UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Aplicación web de carácter social y educativa enfocada al campo de la Entomología

Insect World

José María Toribio Zapata
Tutor: Miguel Ángel Mora Rincón

Mayo 2020

Aplicación web de carácter social y educativa enfocada al campo de la Entomología

Insect World

AUTOR: José María Toribio Zapata
TUTOR: Miguel Ángel Mora Rincón

Grupo GHIA
Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2020

Resumen (castellano)

Insect World es una plataforma web donde la comunidad entomológica podrá interactuar de manera rápida y sencilla. Tras hablar con diferentes apasionados del tema, todos coincidían en que no hay suficientes medios en internet para poder comunicarse y compartir conocimientos.

Creando una aplicación que contenga el mayor número de formas de comunicación, como son el foro, mensajes privados, comentarios en diferentes áreas y chat, se genera un intercambio de información entre usuarios de una manera más fácil y directa sin tener que cambiar de plataforma, ya que por ejemplo, los foros son plataformas aisladas que no cuentan con otro tipo de comunicación.

La calidad de la información compartida es importante, por eso la aplicación contará con un sistema de votos para los comentarios del foro. También se podrá reconocer fácilmente si el usuario que está compartiendo la información es experto o no.

Ya que la aplicación está centrada en entomología, poder contar con imágenes de los insectos es muy importante, por eso la aplicación contará con una galería, una plataforma para subir imágenes y un sistema donde se podrá asociar la imagen de un insecto con la especie a la que pertenecen. Asociar imagen con insecto será solo posible para aquellos usuarios que sean expertos.

Otro punto muy importante son los mapas de distribución. Los entomólogos se guían por las localizaciones geográficas donde fueron capturados los insectos para poder predecir los puntos donde encontrarlos. También, por supuesto, en que época del año es más común encontrarlos.

En definitiva, la aplicación será un punto de referencia para compartir conocimiento, aprender de otros, aportar experiencias, hacer amigos con la misma pasión y tener un registro fiable de la fauna entomológica.

Palabras clave (castellano)

Entomología, Insecto, Determinar, Mapas, Distribución, Gráficas, Comunicación, Social, Comunidad, Aprender, Compartir, Chat, Foro, Familia, Subfamilia, Género, Subgénero, Especie, Subespecie, Imagen, Galería, Coordenadas, Geografía, Zend Framework, RabbitMQ, Docker, Mysql, API, Websockets, Cloudinary, Amazon EC2, Amazon AWS.

Abstract (English)

Insect World is a web application where the entomology community could interact in an easy and rapid way. After talking with different enthusiasts of the topic, all of them agreed that there are not enough tools on the Internet to share knowledge between them.

Developing an application containing as much ways of communication as possible, like it could be forum, private messages, comments in different areas and chat, it generates an exchange of knowledge in an easy and straight way without having to change platform. As an example, forums have a very narrow way of communication because they lack live communication.

Having a voting system ensures that the quality of the information is always reliable. Also it will be easy to recognise if the user sharing the knowledge is an expert or not.

For the entomology community, being able to show and share images is extremely important. Besides, the application will have the ability to associate one image with an specific insect, although this will be possible for expert users only.

Another important point is having distribution maps. Entomologists use the geographical locations where the insects where taken so they can predict where to find them easily in the future. Also in what period of the year is more likely to find them.

To sum up, the application pretends to be a gathering point to share knowledge, learn from others, provide experience, make friends with the same passion and have a reliable record of the entomological wildlife.

Keywords (inglés)

Entomology, Insect, Determine, Maps, Distribution, Graphics, Communication, Social, Community, Learn, Share, Chat, Forum, Family, Subfamily, Genus, Subgenre, Species, Subspecies, Image, Gallery, Coordinates, Geography, Zend Framework, RabbitMQ , Docker, Mysql, API, Websockets, Cloudinary, Amazon EC2, Amazon AWS.

Agradecimientos

La principal persona a la que le quiero agradecer es a mi Padre, por su paciencia, apoyo y por proporcionarme la información necesaria para que el proyecto fuese posible.

También agradecer a mi Madre y a mi Hermana que siempre me han apoyado para que consiguiera terminar el proyecto.

Por último, a mis amigos y a mi novia, que también me han animado para que el proyecto fuese adelante y no me desanimara.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1.Introducción

1.1.Motivación

Mi padre, aunque su profesión fue Arquitecto Técnico, su pasión siempre han sido los insectos. Desde que era pequeño, todas las vacaciones que hacíamos en familia estaban divididas en dos partes. La primera era visitar las principales áreas turísticas de la zona a la que hubiéramos ido. La segunda era perdernos por el campo y acompañar a mi padre en su búsqueda de insectos y en ocasiones, incluso ayudarle en la búsqueda. Desde atravesar glaciares en medio de la montaña hasta adentrarnos en cuevas en medio del monte.

Mi padre lleva dedicándose a los insectos por mucho tiempo, llevándole a determinar varias especies nuevas y a publicar varios libros. Esto ha contribuido a llegar a ser una figura importante dentro del mundo entomológico y una referencia a nivel mundial, en concreto en el estudio de una clase de insectos denominados carábidos (Carabidae).

Durante este tiempo he visto como guardaba información de los insectos en documentos word y tablas excel. El problema es que este sistema no es escalable a la hora de compartirlo y actualizarlo. También he visto como se comunicaba con otros entomólogos mediante cartas y correo. Este tipo de comunicación es efectiva uno a uno, pero no es efectiva uno a muchos. Debido a esto, mi padre siempre ha echado en falta una plataforma en la que poder compartir información de los insectos, añadir las localizaciones geográficas de las capturas, compartir información en tiempo real o semi real o acceder a cualquier otra información relevante. También es muy importante disponer de mapas de distribución para los entomólogos, ya que les proporciona, de una manera fácil y rápida, la ubicación de un determinado insecto en un área geográfica. Mi padre generaba sus propios mapas de distribución, pero una vez terminados y publicados, no podían actualizarse de una forma rápida y sencilla.

Por estos motivos decidí enfocar mi TFG en desarrollar una aplicación de carácter social donde los entomólogos, tanto expertos como aficionados, pudieran interactuar de una manera fácil y directa.

*1.2.*Objetivos

Con este proyecto pretendo crear un lugar donde los amantes de los insectos puedan interactuar, compartir ideas y vivencias y aprender unos de otros.

Los usuarios podrán comunicarse de una manera fácil, incentivando a los usuarios expertos a colaborar, hablando en el foro, o determinando insectos, para que la información sea fiable desde un punto de vista de usuarios más inexpertos.

Mediante fotos, los usuarios tendrán una experiencia más didáctica e intuitiva. Además, los mapas de distribución darán una idea de la localización geográfica de cada insecto.

También se proporcionarán estadísticas de captura de cada insecto con gráficos para saber en qué época del año es más probable capturarlos.

A su vez, el chat proporcionará una comunicación en tiempo real que facilitará el intercambio de información para los usuarios conectados en ese momento, incentivando "quedadas", reforzando a la comunidad entomológica.

1.3. Organización de la memoria

La memoria está organizada de forma progresiva, desde las opciones que se han barajado, hasta las pruebas, y consta de los siguientes capítulos:

- ***Estado del arte*:** En esta sección se proporcionará información sobre qué tecnologías hay disponibles, junto con sus características, ventajas y desventajas.
- ***Análisis y Diseño*:** En esta sección se hablará de los requisitos funcionales y no funcionales del sistema, dando una visión global de lo que hará la aplicación. También se proporcionarán los esquemas y diagramas necesarios para entender la lógica y estructura del sistema.
- ***Desarrollo***: En esta sección se especificará qué tecnologías y herramientas se han elegido para el desarrollo, explicando más a fondo sus características en relación al proyecto.
- ***Integración, Pruebas y resultados:*** En esta sección se hablará de las diferentes técnicas para pruebas empleadas para probar la aplicación y se exponen los resultados.
- ***Conclusiones y Trabajo futuro:*** En esta sección se comentará lo que ha sido la experiencia del desarrollo a nivel profesional y personal. También se expondrán posibles proyectos futuros o grandes mejores y/o adaptaciones del proyecto.

2.Estado del arte

2.1. Frameworks

2.1.1. Entendiendo un Framework

Un framework es un conjunto de librerías donde cada una tiene una finalidad concreta y genérica, facilitando así el desarrollo de una aplicación.
Las razones más importantes para usar un framework son las siguientes:

- **Evitar código repetitivo:** Durante del desarrollo de software, el programador suele encontrarse con escenarios muy similares, como acceso y manipulación de información proveniente de bases de datos, donde se requiere el uso del mismo código. El framework crea clases a más bajo nivel con código reusable.
- **Aplicación de estándares de programación:** La mayoría de los frameworks están basados en (MVC) Modelo-Vista-Controlador, por lo que usarlos te obliga a seguir buenas prácticas.
- **Ahorro y eficiencia de tiempo:** El framework nos ahorra muchas horas de programación ya que implementa códigos muy avanzados que podemos usar sin tener que programarlos nosotros mismo. Normalmente estos códigos nos llevarían muchas horas de trabajo y pruebas.

2.1.2. Frameworks potenciales

Después de investigar que frameworks de PHP había, reduje la lista a ***Zend Framework***, ***cakePHP*** y ***Symfony2.***
Elegí ***cakePHP*** debido a que parece ser un framework fácil de aprender y definitivamente no está demás tener conocimiento de ello para desarrollar pequeñas aplicaciones.
Las principal razón por la que elegí ***symfony2*** y ***Zend Framework***, aunque algo más complejos, es que siguen unos estrictos estándares de programación que PHP aconseja, lo que conlleva a un mejor aprendizaje del lenguaje, siguiendo las mejores prácticas. En los siguientes puntos se profundizará más en sus características.

2.1.3. Zend Framework

Zend framework es una colección de paquetes de PHP que pueden ser usados para desarrollar aplicaciones web, siendo siempre compatible con la última versión de PHP, ya que el framework está desarrollado por PHP. El framework proporciona código totalmente orientado a objetos (Programación Orientada a Objetos). Lo que le hace un framework muy interesante son las siguientes características:

- Al ser orientado a objetos, es muy fácil crear y extender nuevas clases.
- Proporciona un gran número de funcionalidades y módulos, pudiendo usar las que necesites, dejando de lado las demás.

- Aunque es considerado un framework MVC, el modelo (la “M” en MVC) está parcialmente implementado, dando una mayor libertad a la hora de programar nuestro modelo.
- Te da la flexibilidad de poder integrar librerías externas de manera fácil.
- La documentación en general es buena y actualizada y la comunidad es activa.
- Sigue los estándares de programación de PHP, como PSR1 y PSR2.
- El código está probado con pruebas unitarias. (PHPUnit)

2.1.4. CakePHP

CakePHP es un framework de código abierto para el desarrollo rápido de aplicaciones. Al estar desarrollado en PHP te aseguras la compatibilidad con la última versión. La comunidad es activa y tiene librerías internas genéricas para un fácil desarrollo de la aplicación. Entre las características del framework resaltan las siguientes:

- Contiene CRUD que facilita el desarrollo.
- Utiliza arquitectura MVC.
- Fácil integración de AJAX, JavaScript, formularios HTML etc..
- Cache poderosa.
- Generación automática de código.

La pega es que cakePHP es poco flexible y requiere hacer las cosas a su manera. Además el ORM (mapeo objeto-relacional) hace demasiadas consultas a la base de datos, lo que puede ser un problema a medida que el proyecto crezca. También tiene archivos demasiado grandes.

2.1.5. Symfony2

Symfony2 es un framework de cogito abierto para el desarrollo rápido de aplicaciones. Es uno de los frameworks más activos de PHP, aportando soporte y fiabilidad durante el desarrollo. Entre las características del framework resaltan las siguientes:

- El código es muy fiable. Hasta Drupal 8 usa sus librerías.
- Buena documentación y comunidad.
- Compatibilidad con las últimas versiones de PHP.
- Componentes y librerías útiles.

La pega es que symfony2 tiene una curva de aprendizaje bastante alta y no es MVC total, ya que carece de Modelo, haciendo el desarrollo algo más complejo.

2.1.6.Comparación de frameworks

Framework	Comunidad activa	Escalable	POO	MVC	Apps grandes	Apps a largo plazo	Documentación
Zend Framework	*Si*	Si *(Alto)*	*Si*	*Si*	*Si*	*Si*	*Muy buena*
CakePHP	*Si*	S*i* *(Normal)*	*Si*	*Si*	*No*	*No*	*Buena*
Symfony2	*Si*	*Si* *(Alto)*	*Si*	*No del todo*	*Si*	*Si*	*Buena*

Tabla 1. Comparación Frameworks

2.2. Lenguajes

2.2.1. PHP

PHP es un lenguaje de programación ejecutado en el lado del servidor que es normalmente usado para desarrollar aplicaciones web. PHP es una buena opción por las siguientes razones:

- Es gratuito y de "fuente abierta" (open source).
- Es fácil comparado con otros lenguajes como JSP o ASP.
- Tiene una comunidad muy grande y activa.
- Es compatible por defecto con la mayoría de los servidores, teniendo solo que instalar PHP para hacerlo funcionar.
- Tiene actualizaciones periódicas.
- Facilita el uso de bases de datos como MySQL.
- Funciona en cualquier sistema operativo.
- No necesita ser compilado para ejecutarse.
- Incluso Facebook está desarrollado usando PHP.

2.2.2. ASP

El lenguaje de programación ASP, creado por Microsoft, tiene una gran similitud con Visual Basic, por lo que teniendo conocimiento previo facilitaría el desarrollo. Con este lenguaje se pueden crear páginas web dinámicas, aplicaciones y servicios. No es multiplataforma y requiere de un servidor de Microsoft para funcionar.

2.2.3. JSP

Java Server Page (JSP) es un lenguaje con base de Java para desarrollar aplicaciones web. Al usar java se adapta bien a situaciones y lógicas de negocio reales, agilizando el desarrollo de nuestra web. Soporta APIs y el código es completamente

abierto. No posee interfaz y la mayoría del peso recae en el servidor, haciéndolo algo más difícil de mantener.

2.2.4.Comparativa de lenguajes

Lenguaje	PHP	ASP	JSP
Compatible con MySQL	Si	No, necesita drivers	No, necesita drivers
Open Source	Si	No	Si
Integracion con HTML	Fácil	Complicado	Complicado
Compatible con servidores	Compatibilidad muy alta	Necesita servidor dedicado	Compatibilidad alta
Documentación y Comunidad	Muy buena	Normal	Normal
Dificultad de aprendizaje	Baja	Media/Alta	Media/Alta

Tabla 2. Comparación de lenguajes

2.3.HTML y CSS

2.3.1.HTML

HTML es un lenguaje de marcado de hipertexto para crear estructuras de páginas web y que se interpreta en el lado del cliente, es decir el navegador.

HTML viene de "**H**yper**T**ext **M**arkup **L**anguage" donde:

- "HyperText" es el método con el que se navega dentro de la web, haciendo click en texto especial, "hyperlinks", que te direcciona a una página u otra.
- "Markup" hace referencia a los "tags". Tags son códigos especiales que rodean al texto para darle una funcionalidad determinada.
- "Language" viene de lenguaje, ya que tiene sus palabra clave predefinidas que proporcionan una funcionalidad determinada.

2.3.2.Ejemplo HTML

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Título de la página</title>
  </head>
  <body>
    <h1>Esto es una cabecera</h1>
    <p>Esto es un paragrafo</p>
     <a href="www.google.com">Link a la página de google</a>
  </body>
</html>
```

Tabla 3. Ejemplo HTML

2.3.3.CSS

CSS (**C**ascading **S**tyle **S**heets) es un lenguaje simple de diseño para páginas web, que facilita la tarea de hacer una determinada web presentable, es decir, le da estilos a la web. CSS puede controlar como los elementos de una web van a ser presentados. Estos elementos vienen definidos con los tags HTML que comentábamos antes. CSS utiliza dos selectores para determinar a qué elemento aplicar los estilos:

- **Clases**: Se definen con un punto (**.**) y son genéricos, es decir se pueden usar en múltiples elementos. En HTML se usan con el atributo "*class*".
- **Identificadores**: Se definen con una almohadilla o numeral (**#**) y se debe aplicar a un elemento en concreto. En HTML se usan con el atributo "*id*".

CSS es un estándar global en el desarrollo de páginas web, siendo fácil de mantener, compatible con prácticamente todos los dispositivos existentes y previene la repetición de código.

2.3.4.Ejemplo CSS

Siguiendo con el ejemplo de antes, introduciremos estilos a los elementos HTML que definimos:

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Título de la página</title>
     <style>
        .cabecera {
           color: red; /* Texto sera rojo */
           text-align: center; /* El texto estará en el centro */
        }
        #paragrafoId {
           color: blue; /* Texto sera azul */
        }
    </style>
  </head>
  <body>
    <h1 class="cabecera">Esto es una cabecera</h1>
    <p id="paragrafoId">Esto es un paragrafo</p>
     <a href="www.google.com">Link a la página de google</a>
  </body>
</html>
```

Tabla 4. Ejemplo CSS

2.4.MySQL

2.4.1.Base de Datos

Una base de datos es una colección de datos con una estructura determinada que facilita la búsqueda de información específica. Las bases de datos están formadas

normalmente por "*tablas*" que están relacionadas entre si mediante identificadores únicos. Ejemplo de alumnos que van a una determinada universidad:

Tabla Universidades			***Tabla Alumnos***		
Id_universidad	**nombre**		**id_alumno**	**id_universidad**	**nombre**
1	Universidad Autónoma De Madrid		1	3	Pepe
2	Universidad Autónoma de Barcelona		2	3	Juan
3	Universidad Politécnica de Madrid		3	2	Luis

Tabla 5. Ejemplo Tablas relacionales MySQL

En este ejemplo podemos ver que los alumnos "*Pepe*" y "*Juan*" van a la "*Universidad Politécnica de Madrid*" y el alumno "*Luis*" va a "*Universidad Autónoma de Barcelona*".

2.4.2.MySQL

Ya sabemos que es una base de datos, pero, ¿Cómo acceder a los datos que tenemos guardados en ella? Aquí es donde entra en juego MySQL.

MySQL es un sistema de administración de bases de datos basado en SQL (Structured Query Language) que permite extraer, añadir, modificar y borrar información dentro de una Base de Datos.

Un ejemplo sencillo, usando la tabla presentada anteriormente, sería extraer la información de la tabla alumnos que van a la "*Universidad Politécnica de Madrid*":

```
SELECT * FROM alumnos WHERE id_universidad = 3;
```

Otro ejemplo sería escribir una sentencia que nos retorne los alumnos que van a la "*Universidad Politécnica de Madrid*", extrayendo el nombre del alumno y el de la universidad, donde necesitaríamos usar las dos tablas a la vez:

```
SELECT u.nombre, a.nombre
FROM alumnos a
JOIN universidades u ON a.id_universidad = u.id_universidad
WHERE a.id_universidad = 3;
```

2.4.3.Otras alternativas: NoSQL y MongoDB

Al contrario de las *Bases de Datos Relacionales*, donde las tablas tienen unos campos específicos y que deben ser asignados, ya sea con valores por defecto o valores "null", las *Bases de Datos NoSQL* no tienen tablas con una estructura predefinida, sino que se pueden añadir y quitar campos dependiendo de la información que se tenga para cada registro.

Normalmente la estructura usada es definida por JSON, como en el caso de *MongoDB* y se accede a los registros tratándolos como objetos usando JavaScript.
Este tipo de Bases de datos son muy útiles para proyectos grandes donde la probabilidad de que una tabla cambie de estructura es muy alta, dando así una mayor facilidad a la hora de aplicar cualquier cambio, haciéndolo una elección perfecta para proyectos escalables.

2.5.Herramientas para la gestión de entorno de desarrollo

2.5.1. LAMP (desarrollo)

LAMP (***L**inux **A**pache **M**ySQL/**M**ariaDB **P**erl/**P**HP/**P**ython*) es una plataforma de desarrollo web que facilita la instalación de las diferentes aplicaciones involucradas en el desarrollo de un proyecto.
Instalando LAMP me aseguraba tener instalado Apache, MySQL y PHP, que son las herramientas que necesitaba para desarrollar el proyecto, pero la configuración la tenía que hacer manualmente o generando mis propios scripts. También tenía que lidiar con el problema de incompatibilidades entre sistemas operativos.
En definitiva, no me facilitaba una forma rápida y cómoda de instalar mi proyecto en un nuevo servidor, teniendo que ir configurando paso por paso cada área necesaria, con los inconvenientes que esto lleva, ya que siempre hay algún fallo imprevisto.
Debido a esto, decidí integrar la herramienta ***docker*** para estar a cargo del despliegue de la aplicación.

2.5.2. Docker (despliegue)

Docker es un sistema para desplegar aplicaciones de manera fácil y rápida en cualquier entorno de desarrollo. Docker te ahorra el proceso de instalación cada vez que necesitas desplegar tu aplicación en un servidor nuevo o en una máquina local. También te ahorra el tener que preocuparte por incompatibilidades relacionadas con sistemas operativos, ya que puede funcionar como una máquina virtual con base de Linux.

Docker está formado por “contenedores” o “containers”, que son creados usando “imágenes” o “images”.

- **Images**: Las “images” son archivos que definen lo que tendrá un “container”. Por ejemplo, se puede tener una imagen que instalará una versión específica de php con una configuración determinada.
- **Containers**: Los “containers” son instancias creadas a partir de las “images”. Se pueden generar tantos “containers” como se quiera. Siguiendo el ejemplo anterior, tendríamos un entorno de desarrollo corriendo con una versión específica de php con una configuración determinada.

Un símil que se puede usar puede ser que la “imagen” es como una receta de un pastel, y el “container” es el pastel en sí.

En este caso, el proyecto consta de tres “containers”. Uno donde está instalado php y el código fuente, otro donde está instalado mysql, y por último uno donde está instalado rabbitMQ.
Para relacionar los tres “containers" y que puedan interactuar entre ellos use la herramienta **docker-compose.**

2.5.3. Otras alternativas: Vagrant (despliegue)

Vagrant es otra potente herramienta para automatizar el despliegue de aplicaciones. Vagrant usa virtualización, mientras que docker usa contenedores. La diferencia entre ellas es que la virtualización usada en Vagrant utiliza máquinas virtuales con sistemas operativos enteros e independientes para cada una de ellas, sin embargo Docker utiliza un único sistema operativo como base, y desde ahí empieza a crear contenedores independientes, pero que comparten el núcleo del sistema operativo, en este caso linux.
Por esta razón decidí usar Docker en lugar de Vagrant.

2.6. RabbitMQ (sistema de colas)

RabbitMQ es un sistema de mensajes y colas. RabbitMQ puede usarse para diferentes fines pero el más común es el envío de información (mensajes) de forma fiable, asegurando que la información llega siempre a su destino. Para asegurar la entrega, RabbitMQ se ayuda de un sistema de colas. Una cola es un buffer donde el mensaje se almacena hasta que es enviado al destinatario. Si algo falla, el mensaje es reenviado. Cada cola está asociada a un consumidor o “consumer”, que es una pieza de código que espera la llegada de los mensajes y hace las operaciones deseadas con la información. Si el “consumer” no está corriendo, el mensaje se almacena en la cola sin perder nada de información.

En el proyecto utilizo RabbitMQ para enviar emails a los usuarios, informándolos de cambios relacionados con ellos, como nuevos mensajes privados. Gracias a esto me aseguro que el usuario recibirá el correo y podrá estar al día de su contenido.

Figura 1. Ejemplo de cola RabbitMQ

En la figura 1 vemos como en “P” se envía a la cola el mensaje con los datos necesarios para enviar el email. Cuando “C” está preparado (el “consumer” está corriendo) la cola procesa el mensaje y “C” ejecuta el código, generando el email con los información proporcionada en el mensaje y envía el email.

2.7. Kafka (sistema de colas)

Kafka es otro sistema de mensajería y transmisión de datos basado en colas. Es un software de código abierto, y como rabbitMQ solventa los principales problemas entre el lado que envía información y el lado que la procesa. Entre ellos evita que el receptor se vea saturado por el emisor. También asegura que un mensaje enviado pero no procesado correctamente pueda ser reubicado en la cola para ser procesado posteriormente. Aunque Kafka es un buen sistema de colas, es menos flexible que rabbitMQ, ya que proporciona menos libertad a la hora de crear colas dependientes de otras, colas de proceso de mensajes de una a muchas, etc…

2.8. Websockets (protocolo bidireccional)

Websockets es un protocolo para crear una conexión rápida de dos direcciones mediante el navegador y el servidor. Websockets nos ayuda a crear una aplicación dinámica en tiempo real, sin la necesidad de que el usuario interactúe con el navegador, como podría ser con AJAX. Esto es útil para actualizar contenido en la aplicación en tiempo real, o incluso para crear un chat, como es el caso de este proyecto.

WebSockets nos proporciona una comunicación verdaderamente instantánea en la web, haciendo que los sitios web se vean y se sientan más como aplicaciones locales.

Las ventajas son las siguientes:

- Los usuarios disfrutan de una experiencia fluida e ininterrumpida, ya que su navegador se comunica constantemente con el servidor, entregando contenido nuevo inmediatamente en lugar de cuando se solicita.
- Las empresas ven una mayor satisfacción y compromiso del cliente a medida que los clientes disfrutan de una experiencia más dinámica y fluida.

Figura 2. Comparación Webscokets y HTTP

En el proyecto utiliza Websockets para crear una sistema de chat integrado en la aplicación. Los usuarios registrados pueden conectarse al chat e interactuar con otros usuarios de manera rápida y en tiempo real, lo que hace que la aplicación tenga un carácter más social.

2.9. HTTP/2 (protocolo bidireccional)

HTTP/2 es un protocolo para establecer conexiones bidireccionales que usa las mismas cabeceras que el protocolo HTTP/1.X. HTTP/2 proporciona “multiplexed

streams", que quiere decir que se aprovecha la misma conexión para hacer múltiples peticiones.
Aunque HTTP/2 es un buen protocolo, tiene la desventaja de que el lado del cliente y el servidor son asimétricos, es decir, no pueden hacer las mismas peticiones y respuestas. Esto es diferente con websockets, donde los dos lados son iguales, dándole una mayor flexibilidad.

3.Análisis y diseño

3.1.Actores

Habrá cuatro tipos de usuarios que podrán interactuar con la aplicación, donde cada uno de ellos tendrá diferentes permisos:

- Usuario **invitado/no registrado**: Tendrá acceso a las áreas comunes, pero no podrá interactuar con otros usuarios.
- Usuario registrado **básico:** Tendrá acceso a todas las áreas y podrá interactuar, pero tendrá restricciones a la hora de determinar insectos.
- Usuario registrado **experto:** Tendrá acceso a todas las áreas y podrá interactuar, pero no tendrá restricciones a la hora de determinar insectos.
- Usuario registrado **administrador:** Tendrá acceso a todas las áreas y podrá interactuar, no tendrá restricciones a la hora de determinar insectos y podrá administrar información de usuarios.

3.2.Requisitos funcionales

3.2.1.Especificaciones usuario invitado

Estas especificaciones serán para los usuarios invitados , es decir, que no se hayan identificado en el sistema.

3.2.1.1.Registro de usuario

El usuario invitado podrá registrarse en el sistema mediante un formulario en la sección "Registro". El formulario contará con dos pasos.

En el primer paso se le pedirá al usuario que introduzca los siguientes campos:

- Nombre (obligatorio).
- Apellidos (obligatorio).
- Alias (obligatorio, posteriormente será usado para "loguearse").
- Contraseña (obligatorio, posteriormente será usado para "loguearse").

- Confirmación de contraseña (obligatorio, posteriormente será usado para registrarse).

Si todo ha ido bien en el paso uno, se procede al paso dos:

- Correo (Se usará para enviar notificaciones. No visible por otros usuarios) (obligatorio).
- Sexo (obligatorio) (Hombre/Mujer).
- Ciudad (obligatorio).

El usuario podrá navegar entre el paso uno y paso dos libremente, sin perder los datos introducidos. Una vez registrado, se direccionará al usuario a la página de **"Login".**

3.2.1.2.Acceso a la aplicación mediante credenciales

El usuario invitado (una vez registrado), podrá acceder al sistema mediante la página de **"Login".** En esta sección se le pedirá al usuario su **Alias** y su **contraseña.** Si los datos son correctos el usuario habrá entrado al sistema satisfactoriamente.

3.2.2.Especificaciones comunes a usuario invitado y usuario registrado básico y experto

En esta sección enumeraremos las especificaciones comunes a los usuarios **invitados** y a los usuarios registrados **básico** y **experto**. Se resaltarán las áreas restringidas para cada usuario según su nivel de privilegios en la aplicación.

3.2.2.1.Cerrar la sesión

El usuario, una vez registrado y "logueado", podrá cerrar la sesión en cualquier momento, independientemente del área en el que se encuentre.

3.2.2.2.Acceso a todas las discusiones del sistema (relacionado con el foro)

El sistema facilitará una lista con todas las discusiones que haya en el sistema hasta ese momento. Las lista se mostrará en forma de tabla, conteniendo las siguientes columnas:

- **Discusiones**: Habrá una imagen, que dirá si la discusión ha sido visitada después de la última modificación (blanca) o por el contrario no ha sido visitada o tiene nuevos topics y/o mensajes (roja), seguida de un nombre y una descripción de la discusión. El nombre será un link que el usuario podrá pinchar para acceder a la discusión (ver apartado 3.2.2.3). La imagen será siempre roja para usuarios invitados.
- **Topics**: Número de topics que la discusión posee.
- **Mensajes**: Número de mensajes que la discusión posee.
- **Últimо mensaje**: Muestra el nombre del usuario que escribió el último mensaje, junto con la fecha de creación. Se puede acceder al perfil de ese usuario y al mensaje en cuestión haciendo click sobre ellos.

3.2.2.3.Acceso a todos los topics del sistema (relacionado con el foro)

El sistema facilitará una lista con todos las topics que haya en el sistema hasta ese momento. Las lista se mostrará en forma de tabla, conteniendo las siguientes columnas:

- **Topics**: Habrá una imagen, que dirá si el topic ha sido visitada después de la última modificación (blanca) o por el contrario no ha sido visitado o tiene nuevos mensajes (roja), seguida del nombre del topic, el nombre del usuario que creó el topic y la fecha de creación. El nombre del topic será un link que el usuario podrá pinchar para acceder a los mensajes del topic (ver apartado 3.2.2.6) y el nombre del usuario creador del topic será un link para acceder al perfil del usuario. La imagen será siempre roja para usuarios invitados.
- **Respuestas**: Número de respuestas que el topic posee.
- **Visitas**: Número de visitas que el topic posee.

3.2.2.4.Búsqueda y filtrado de los topics

El usuario podrá buscar fácilmente topics, filtrando por cualquier palabra que aparezca en el topic (en la tabla). Los resultados de la búsqueda aparecerán en la pantalla a medida que el usuario teclea su opción de filtrado.

3.2.2.5.Añadir nuevo topic al sistema

El usuario registrado **básico y experto** tendrán los permisos necesarios para crear un nuevo topic para una discusión determinada, mediante un formulario.

El formulario constará de tres campos:

- Título del topic (obligatorio).
- Contenido del primer mensaje (obligatorio).
- Botón de envío “Nuevo Topic”.

Una vez creado, se direccionará a la página del nuevo topic, donde se mostrará como primer mensaje.

3.2.2.6.Acceso a todos los mensajes del sistema (relacionado con el foro)

Una vez el usuario selecciona un topic, se direccionará a la sección de mensajes para ese topic. Se mostrarán todas las respuestas a ese topic con información relevante de cada usuario:

- Nombre de usuario.
- La posición numérica en la que fue posteado el mensaje para ese topic.
- Foto de perfil del usuario.
- Fecha en el que fue posteado el mensaje.
- Número de posts para ese usuario.
- Cuando se registró el usuario.

- De donde es el usuario.
- Número de votaciones positivas.
- Botón para votar positivamente.

3.2.2.7.Búsqueda y filtrado de los mensajes del topic (relacionado con el foro)

El usuario podrá buscar fácilmente mensajes del topic, filtrando por cualquier palabra que aparezca en el mensaje del topic (en la tabla). Los resultados de la búsqueda aparecerán en la pantalla a medida que el usuario teclea su opción de filtrado.

3.2.2.8.Añadir nuevo mensaje en un topic

El usuario registrado **básico y experto** tendrán los permisos necesarios para crear un nuevo mensaje para un topic determinado, mediante un formulario.

El formulario constará de dos campos:

- Contenido del mensaje (obligatorio).
- Botón de envío.

La aplicación añadirá el nuevo mensaje al topic al presionar el botón de envío y se direccionará a la página de mensajes para ese topic.

3.2.2.9.Votar positivamente un mensaje en el topic (relacionado con el foro)

El usuario registrado **básico** y **experto** tendrán los permisos necesarios para votar positivamente un mensaje de un topic determinado, mediante un botón. Esto incrementará un contador que reflejará la utilidad del mensaje para otros usuarios. Cada usuario solo podrá votar una vez cada mensaje.

3.2.2.10.Listado de usuarios registrados

El sistema facilitará a los usuarios la posibilidad de ver un listado con todos aquellos usuarios registrados. La lista mostrará el *nombre (se podrá acceder al perfil haciendo "click")*, *fecha de activación (alta)*, *web* (si se ha proporcionado) y *foto de perfil* en miniatura de cada usuario (si no se ha proporcionado, se mostrará la foto default). También se podrá ordenar la lista por columnas y buscar por nombre

3.2.2.11.Perfil de usuario

El usuario puede acceder a su propio perfil personal o al de otros usuarios del sistema. Dependiendo de si es nuestro propio perfil, o estamos de visita en otro, nos encontramos con diferentes escenarios:

Datos comunes:

- Nombre de acceso del usuario en el título del panel, en la parte superior.

- Foto de perfil subida por el usuario, o una foto por defecto si el usuario no subió foto aun.
- Nombre real del usuario.
- Fecha en el que el usuario se registró.
- Fecha de última vez que el usuario se logueó.
- Correo electrónico del usuario, si el usuario lo introdujo.
- Web personal del usuario, si el usuario la introdujo.
- Ciudad en la que reside el usuario.
- Número total de mensajes en el foro.
- Número total de logueos.
- Sexo del usuario, masculino o femenino.
- Pestaña "Topics", donde se muestran los topics en los que ha participado el usuario. Se podrá ver el número de respuestas, número de visitas y fecha de creación de cada topic, así como un link directo al topic en cuestión.
- Pestaña "Mensajes", donde se muestran todos los mensajes que el usuario ha escrito hasta ahora.
    - El usuario podrá buscar lo que necesite dentro de la pestaña tecleando lo que desea buscar.
    - El usuario podrá navegar al topic donde aparece el mensaje
- Pestaña "Fotos a determinar", donde se muestran las fotos que ha subido el usuario para ser determinadas
    - Al presionar la foto el usuario será direccionado al perfil del insecto.

Funcionalidades cuando se visita el perfil propio:

- Subir y/o actualizar foto de perfil. El usuario tendrá dos opciones para hacer esto:
    - Presionando el botón "Subir foto" y eligiendo la foto que desea subir. El botón tendrá un mensaje informativo al pasar el ratón por encima. La foto se actualizará automáticamente después de subirla.
    - Arrastrando una foto al cuadrado con el texto "O ARRASTRA LA FOTO AQUÍ". La foto se subirá directamente una vez se suelte la foto al cuadrado y se actualizará automáticamente.
- Actualizar información personal del usuario. Al presionar el botón "Editar perfil" una ventana emergente con un formulario aparecerá. El botón tendrá un mensaje informativo al pasar el ratón por encima. El usuario podrá modificar los siguientes campos:
    - Nombre del usuario.
    - Apellidos del usuario.
    - Correo electrónico del usuario
    - Web personal del usuario.
    - Sexo del usuario.

- Ciudad en la que reside el usuario.
- Botón “Aplicar cambios” que guardara los cambios. Los cambios se verán reflejados automáticamente sin necesidad de refrescar la página.

Funcionalidades cuando se visita otro perfil:

- El usuario podrá ***enviar un mensaje privado*** al usuario pinchando el botón “Mensaje privado”. Una ventana emergente con un formulario aparecerá. Contendrá los siguiente campos:
  - Un área para añadir el contenido del mensaje.
  - Botón “Enviar” que enviará el mensaje al usuario deseado.

3.2.2.12.Aviso de mensajes privados

La aplicación avisará al usuario, una vez “logueado”, de si tiene nuevos mensajes privados, añadiendo el número de mensajes nuevos en el menú, junto al link “mensajes privados” y entre paréntesis. El usuario también será notificado del nuevo mensaje mediante un email. El email contendrá un link de un solo uso, que direccionará al usuario al mensaje, logueandolo directamente en la aplicación.

3.2.2.13.Registro de usuario

El usuario, una vez registrado y “logueado”, podrá acceder a la sección de “Mensajes Privados”, donde se mostrará una lista de los usuarios con los que se han realizado intercambios de mensajes. La lista será en forma de tabla con los siguientes campos o columnas:

- **Mensaje de:** Nombre del usuario con el que se está intercambiando mensajes.
- **Experto:** Si es experto o no.
- **Mensajes recibidos:** Número de mensajes intercambiados más un link con la palabra “leer” que direccionará al usuario a la sección donde podrá leer los mensajes.

La fila que contenga mensajes sin leer será de distinto color a las otras. El usuario podrá también además filtrar en la lista mediante búsqueda con palabras clave.

3.2.2.14.Hilo de mensajes con otro usuario

El usuario podrá ver todos los mensajes intercambiados con otro usuario, incluido los suyos. Se mostrarán ordenados por orden de fecha, de más nuevos a más viejos. Habrá un formulario para añadir una contestación para continuar el hilo de la conversación. El formulario constará de dos campos:

- Cuadro de texto para escribir el mensaje (obligatorio)
- Botón para enviar el mensaje

El mensaje será añadido automáticamente sin necesidad de refrescar la página. El usuario destinatario recibirá el mensaje y podrá leerlo siguiendo la misma lógica explicada anteriormente.

3.2.2.15.Galería de imágenes de insectos a determinar

Los usuarios, tanto invitados como registrados, podrán acceder a la galería de insectos a determinar. En esta sección se mostrará una imagen previa de cada imagen subida por cada usuario con la siguiente información:

- Título
- Autor de la foto
- Fecha
- Si el insecto está determinado o no.

La página mostrará un máximo de doce imágenes previa. El usuario podrá ver las siguiente imágenes pulsando en el botón "Siguiente". También podrá ir hacia atrás pulsando el botón "Atrás". Se podrá pulsar en cada imagen, donde el usuario será direccionado al perfil asociado a esa imagen.

3.2.2.16.Subida de imágenes de insectos a determinar

Esta sección es solo accesible para **usuarios registrados**. Los usuarios podrán subir la imagen del insecto qué desean determinar mediante un formulario. El formulario consta de los siguientes campos, donde todos son obligatorios:

- Área para seleccionar la imagen del insecto.
- Título de la foto (si se deja en blanco, el título será "A determinar" por defecto).
- Ubicación donde fue tomada la imagen.
    - El usuario tecleará la ubicación y el sistema le dará todas las posibles opciones a medida que escribe.
    - Una vez seleccionada la ubicación, el mapa posicionado a la derecha del formulario será actualizado con la nueva ubicación automáticamente.
    - Los campos "Latitud" y "Longitud" serán rellenados automáticamente también.
- Breve descripción de la foto.
- Fecha en la que fue tomada la foto.
    - El usuario podrá seleccionar una fecha (no menor que el día actual) en un calendario.
- Latitud.
- Longitud.
- El usuario tendrá un botón con la opción de actualizar el mapa y la ubicación introduciendo las coordenadas (latitud/longitud).
- El usuario tendrá la opción de auto rellenar la ubicación, latitud y longitud pulsando sobre el mapa en la ubicación deseada.

3.2.2.17.Perfil insecto a determinar

Cada insecto que es subido al sistema, determinado o no, tiene su propio perfil donde contiene información relevante del insecto. Según que usuario entre al perfil, se podrán hacer y/o ver diferentes cosas.

Áreas comunes a todos los usuarios:

- Información sobre la captura y foto del insecto
  - Título de la foto.
  - El autor de la foto.
  - Si está determinado o no. Si lo está, se mostrará el nombre científico del insecto y el nombre del usuario que lo determinó con un link a su perfil.
  - Fecha de captura.
  - Lugar de captura.
  - Latitud.
  - Longitud.
  - Breve descripción de la foto.
- Mapa con el punto donde se encontró el insecto.
- Acceso al mapa de distribución (mirar siguiente punto).
- Comentarios.

Usuarios registrados básicos y expertos podrán añadir comentarios al perfil.

Usuarios registrados expertos podrán determinar el insecto

- Podrán buscar por el insecto en la base de datos y asignárselo al perfil.
- Si no se encuentra en la base de datos, podrán añadir un nuevo insecto al sistema y lo asignaran al perfil.

3.2.2.18.Mapa de distribución insecto

El sistema generará un mapa de distribución basado en el número de capturas registradas, tanto por usuarios del sistema, como capturas oficiales extraídas de una base de datos externa mediante una API. El mapa mostrará una "nube" con puntos calientes, donde el color rojo determina mucha actividad y el color verde poca actividad. Adicionalmente se proporcionará una gráfica donde se podrá ver la relación entre el número de capturas y el mes en el que fue capturado el insecto.

3.2.2.19.Chat en tiempo real

Una vez registrado, el usuario será conectado a una chat automáticamente. El chat mostrará los mensajes de otros usuarios, y los usuarios conectados en ese momento. Se podrá cerrar sesión en cualquier momento haciendo click en "Cerrar Sesión", evitando que otros usuarios vean tu conexión. De igual forma, se podrá volver a conectar al chat haciendo click en "Entrar".

3.3.Requisitos no funcionales

En esta sección se enumerarán algunas de las características técnicas del sistema relacionadas con el rendimiento, almacenaje, usabilidad, fiabilidad y confidencialidad, entre otras.

3.3.1.Procesador y almacenamiento de imágenes en la nube: Cloudinary

Cloudinary es una herramienta online para almacenar y procesar fotos y vídeos directamente en la nube. Esta herramienta de terceros facilita la gestión de fotos en la aplicación, permitiendo almacenar fotos fácilmente y modificar sus tamaños y formas, mostrándolas directamente al usuario usando la API que proporciona la herramienta.

3.3.2.Sistema de colas para mandar emails: rabbitMQ

La aplicación utiliza un sistema de colas para mandar emails, asegurándose que todos los emails son enviados correctamente. Este sistema, que fue explicado en el punto ***2.6 RabbitMQ***, proporciona una alta fiabilidad de que los emails lleguen al usuario final sin ningún tipo de problema. En el caso de que el sistema falle, el email queda almacenado hasta que el sistema vuelva a funcionar, y este es enviado.

3.3.3.Fiabilidad y tiempo de actividad del servidor

La aplicación estará alojada en un servidor EC2 AWS de Amazon, con una instancia de tamaño t2.micro y un porcentaje de actividad de 99.95%. Este tipo de servidor nos proporciona una alta fiabilidad con un **tráfico** razonable.

3.3.4.Encriptación de datos sensibles

El sistema encriptará datos sensibles como las contraseñas. Esto asegura que la confidencialidad de cada cuenta quede asegurada, siendo solo el usuario final el que sabe la contraseña real.

3.4.Casos de uso

3.4.1.Casos de uso del apartado foro

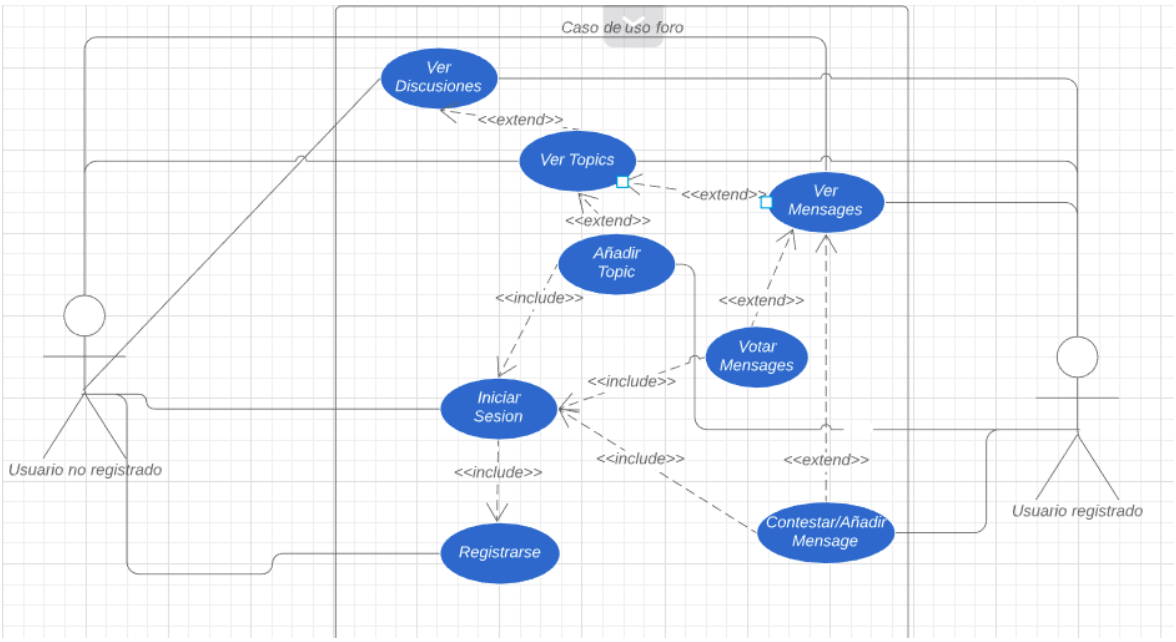


Figura 3. Casos de uso del foro

3.4.2.Casos de uso de interacción entre usuarios

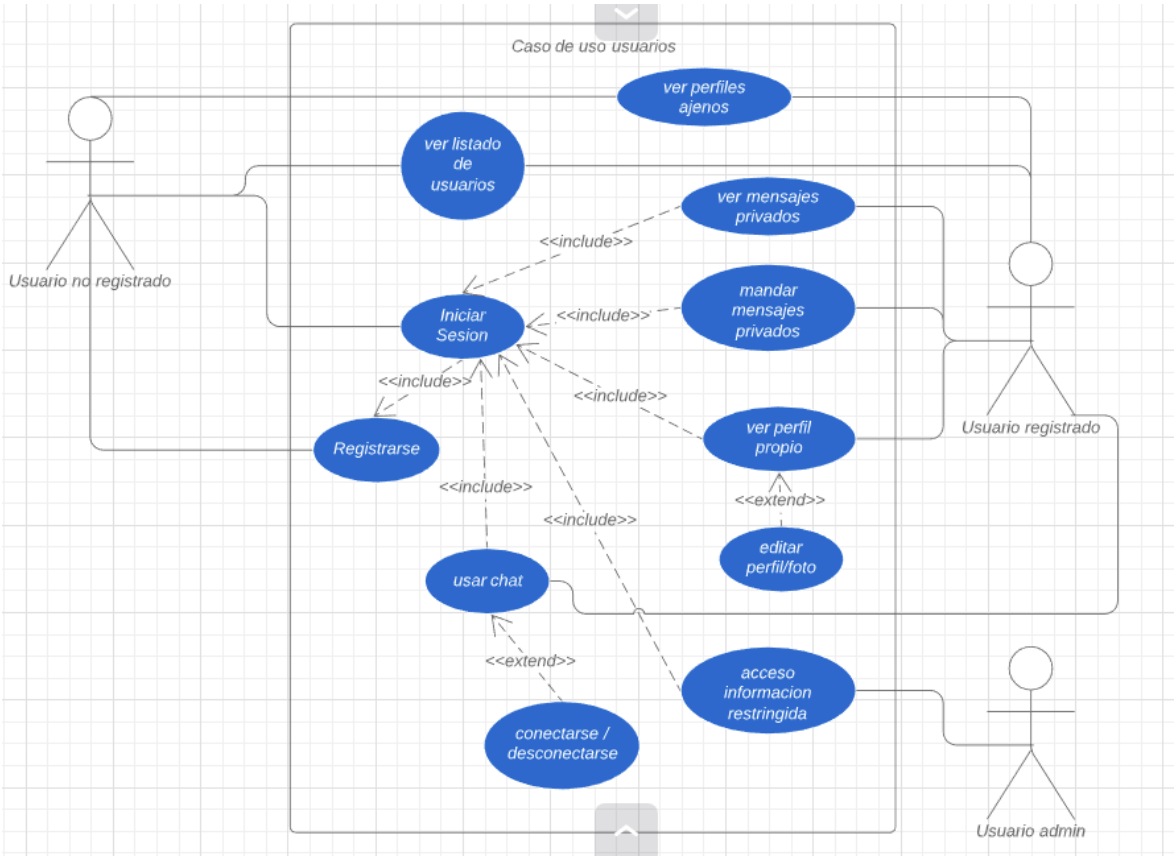


Figura 4. Casos de uso interacción usuarios

3.4.3.Casos de uso determinar insectos

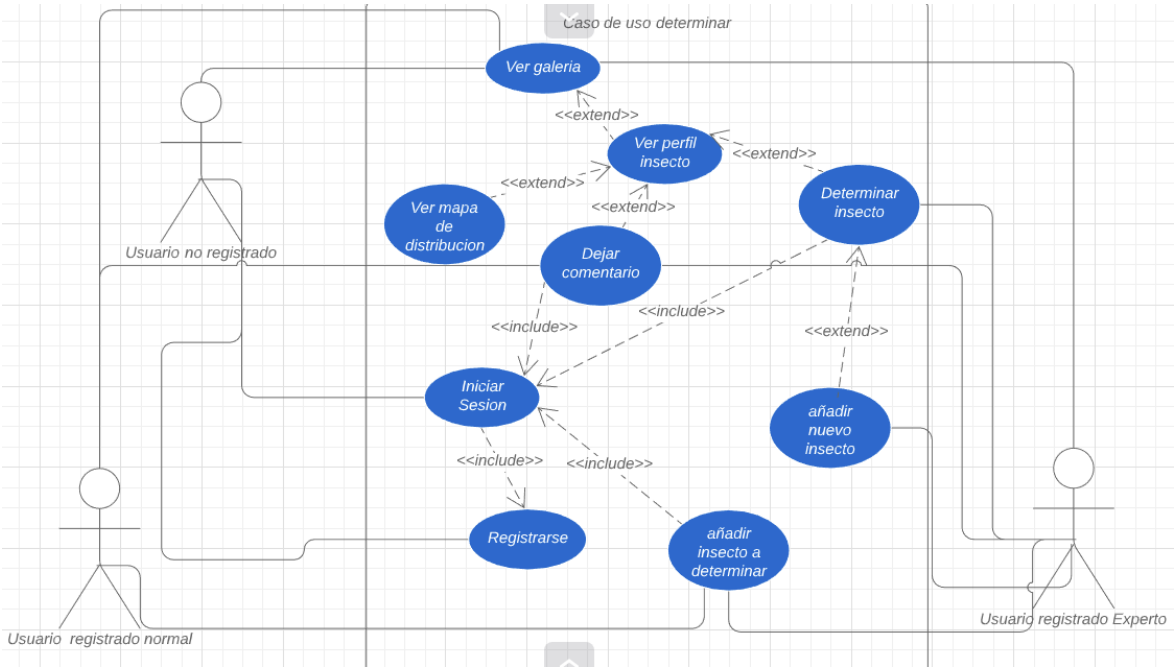


Figura 5. Caso de uso determinar insectos

3.5. Modelo UML entidad-relación de la base de datos

3.5.1.Tablas

La base de datos consta de 18 tablas que interacciones entre sí para proveer a la aplicación de un almacenaje robusto de datos:

- ***auto_login_url:*** Almacena la información para el link de auto logueo único y de un solo uso.
- ***chat_counter:*** Número total de mensajes en el chat.
- ***chat_log:*** Contenido de los mensajes en el chat.
- ***Determinar:*** Información del insecto proporcionado por el usuario para ser determinado.
- ***determinar_comentario:*** Mensajes públicos de un insecto a determinar.
- ***Discusion:*** Temas principales del foro.
- ***distribucion_insecto:*** Coordenadas de captura de los insectos
- ***imagen:*** Imágenes del sistema. Tanto de usuarios como generales.
- ***Insecto:*** Estructura con la información de un insecto.
- ***mensaje:*** Mensajes del foro.
- ***mensaje_privado:*** Mensajes privados entre usuarios.
- ***mensaje_privado_contenido:*** Contenido de los mensajes.
- ***mensaje_privado_map:*** Relación entre los dos usuarios involucrados en los mensajes.
- ***read_discusion:*** Determina si una discusión en el foro ha sido ya leída por un usuario.
- ***read_topic:*** Determina si un topic en el foro ha sido ya leído por un usuario.
- ***Topic:*** Contenido de los topics del foro.
- ***Usuario:*** Información del usuario.
- ***vote_log:*** Determina si un mensaje del foro ha sido votado o no y por quien.

3.5.2.Diagrama UML

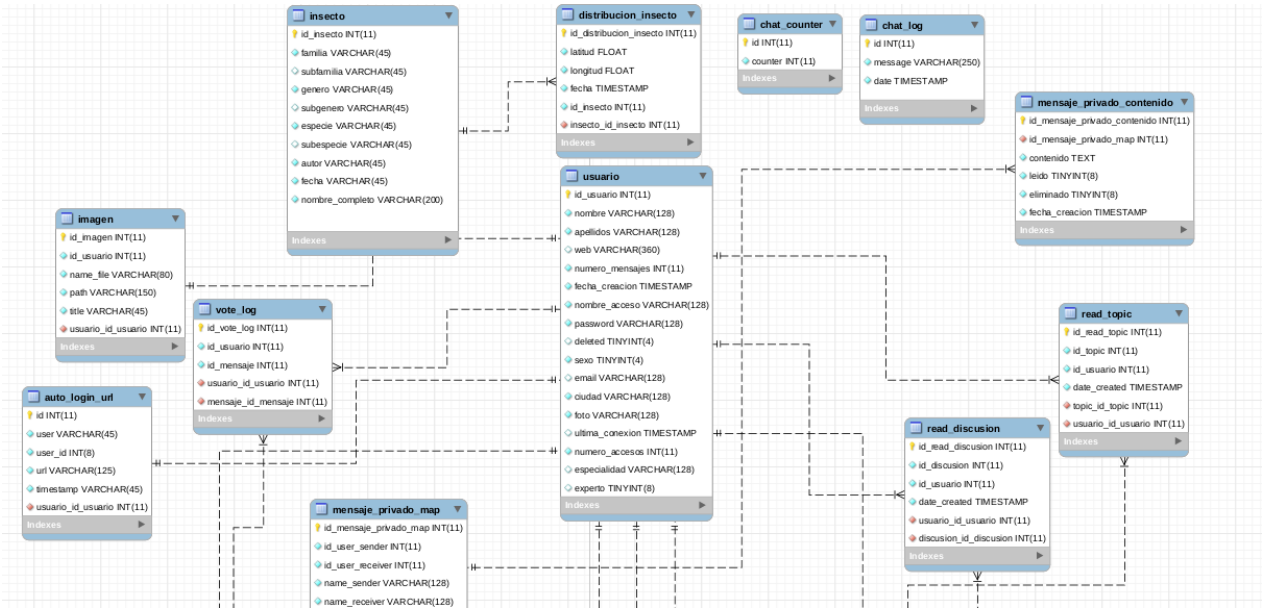


Figura 6. Modelo UML entidad-relación parte 1

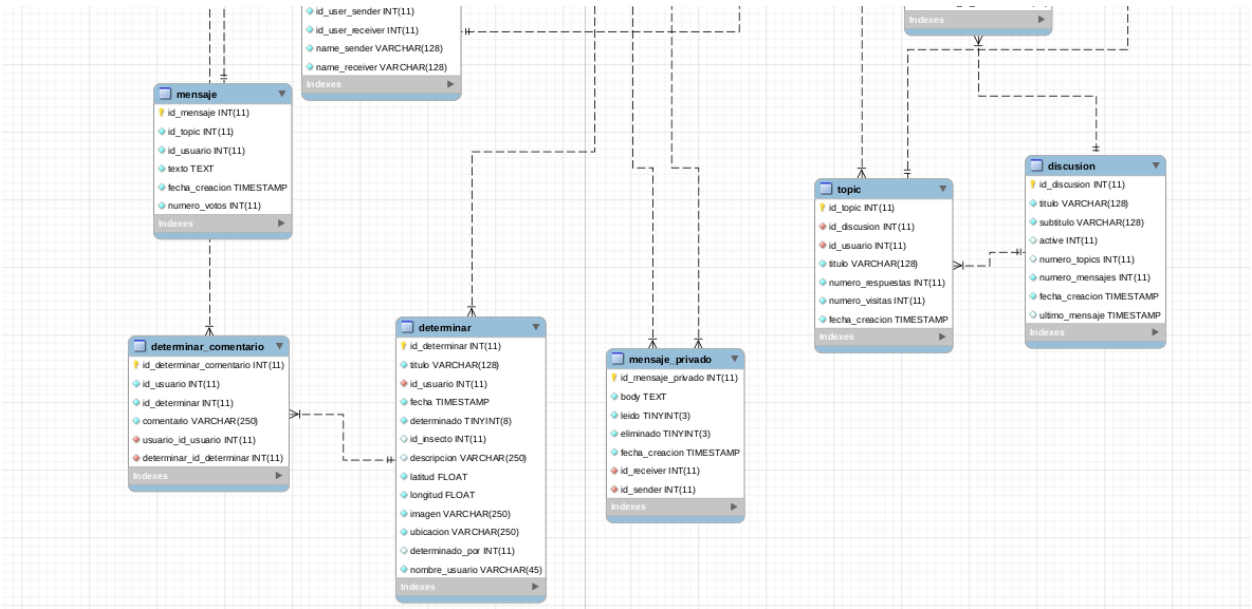


Figura 7. Modelo UML entidad-relación parte 2

3.6.API externa para conseguir datos reales sobre insectos

Después de investigar sobre como poder adquirir datos reales y fiables de insectos, encontré la base de datos ***GBIF.***

GBIF es una infraestructura de datos abiertos fundada por los gobiernos del mundo que proporciona, a quien lo necesite, cualquier información sobre cualquier ser vivo registrado en la tierra. Dispone de una API abierta de la que se pueden extraer los datos que se necesiten. Un ejemplo es el siguiente:

```
http://api.gbif.org/v1/occurrence/search?
country=ES&orderKey=1470&familyKey=3792&limit=300&offset=0
```

Los parámetros del ejemplo sirven para filtrar el resultado:

- ***country***: determina el país de procedencia del ser vivo.
- ***orderKey:*** determina qué tipo de ser vivo es, en este caso invertebrados (insectos).
- ***familyKey:*** determina qué tipo de invertebrados, en este caso son carábidos.
- ***limit***: determina el número de resultados que devuelve.
- ***offset***: determina desde donde empieza a buscar, para así no repetir resultados.

Con la ayuda de un script que se comunicara con la API, podremos rellenar la base de datos con insectos reales, guardando sus nombres, localización y otros datos necesarios, disponiendo así de datos reales para darle una imagen más robusta y segura a la aplicación.

3.7.Tecnologías seleccionadas para el desarrollo

Después de analizar las posibles alternativas tecnológicas para el proyecto, se ha llegado a la conclusión de que las mejores opciones para el desarrollo son las siguientes:

En el lado del servidor usaremos el ***Framework Zend***, que es un framework de PHP basado en Modelo-Vista-Controlar (***MVC***). Esto nos dará una robustez en el código y una mayor facilidad de desarrollo. Para el almacenaje y manipulación de información usaremos una base de datos ***MySQL.*** Aparte de ser una de las base de datos más potentes, el ***Framework Zend*** ya viene con librerías integradas que nos facilitan la tarea de conexión a la base de datos, petición de datos, etc…

En el lado del cliente, usaremos ***HTML*** y ***CSS*** para el diseño básico. También usaremos ***JavaScript*** que le dará un toque más dinámico. A su vez, usaremos ***AJAX*** para comunicarnos con el servidor en un segundo plano, dando al usuario una sensación de uso en tiempo real.

El chat se implementará usando ***WebSockets***, proporcionando una conexión bidireccional en tiempo real entre usuarios.

Para asegurarnos que las peticiones al servidor en casos especiales, como por ejemplo envío de emails, no se pierden y pueden ser relanzadas en caso de error, usaremos el sistema de colas que proporciona la librería ***RabbitMQ***.

En la parte del desarrollo de más bajo nivel tendremos diferentes tecnologías.
La aplicación vivirá en una instancia ***EC2*** que proporcionan los ***Servicios Web de Amazon (AWS)***. Podremos conectarnos a la instancia remotamente usando ***SSH*** y podremos cambiar o configurar lo que necesitemos.
Para el despliegue de la aplicación usaremos la herramienta ***Docker***, facilitando la tarea de despliegue en máquinas nuevas o cuando haya cambios en el código. ***Docker*** nos dará la habilidad de tener cada parte de la aplicación en diferentes *contenedores*, siendo más fácil modificar partes especificas sin afectar otras.

4.Desarrollo

El desarrollo del proyecto ha constado de diferentes fases, que se han ido adaptando a las necesidades que han ido surgiendo.

4.1. Preparación del entorno de desarrollo

Para poder empezar a desarrollar el proyecto, lo primero que se hizo fue instalar y configurar la máquina local con **LAMP**.
El conjunto de tecnologías de **LAMP** (**Linux, Apache, MySQL y PHP**), proporciona todas las herramientas necesarias para poder desarrollar la aplicación desde cero. La máquina local necesita tener **Linux** como sistema operativo. El servidor para emular peticiones y respuestas de manera local se logra mediante el servidor **Apache**. La instalación es rápida y sencilla.
Para poder usar el **Framework Zend** se instalaron los paquetes necesarios de **PHP**.
Así mismo, se instalaron los paquetes necesarios para poder usar la bases de datos de **MySQL** y poder interactuar con ella posteriormente, tanto vía código como vía cliente (**Mysql Workbench**).
Por último, se instaló el **IDE** necesario, en este caso **NetBeans**, para poder programar el código y se configuró el repositorio GitHub, con su correspondiente clave **RSA** para conectar vía **SSH**, para poder guardar y descargar el código remotamente y poder trabajar sobre la marcha con diferentes versiones usando **Git**.

4.2. Implementación del código

Para la implementación del código se siguió una ***metodología ágil***. Con este método se consigue un desarrollo que está preparado para sufrir cambios rápidos, ya que no le da una importancia excesiva a lo que se había programado, dándole una flexibilidad muy grande a la hora de introducir mejoras, funcionalidades de última hora o corrección de errores.

Podemos separar el desarrollo en cuatro grandes fases: *Foro, Usuarios, Insectos* y *Chat.*

4.2.1.Foro

El foro sigue una lógica de un foro clásico, con discusiones, topics y mensajes de un topic determinado.

Las principales dificultades para el foro fueron las relaciones entre un usuario y sus interacciones con el foro y los demás usuarios. Se tuvo que pensar bien el diseño de la base de datos y sus relaciones para hacerlo lo más óptimo y escalable posible.
Otra dificultad fue llevar una cuenta visual de lo que un usuario ha visitado o no ha visitado (discusiones, topics y mensajes) en tiempo real.

4.2.2.Usuarios

En esta fase se ha implementado todo lo relacionado con los usuarios, desde el registro y "logueo", hasta el perfil de usuario.

Una de las principales dificultades fue lidiar con los distintos niveles de permisos entre los distintos tipos de usuarios registrados. Se tuvo que implementar de manera global un sistema que reconociera el usuario registrado en ese momento, para así poder ver si lo que pide está dentro de su rango de acción. Esto evita accesos directos a funcionalidades directamente usando la URL sin estar registrado o tener suficientes permisos.

Otra dificultad fue implementar la relación entre usuarios para mensajes privados, junto con un aviso por email mediante el sistema de colas ***RabbitMQ***.
El script que levanta la cola de mensajes de ***RabbitMQ*** se caía de manera aleatoria, posiblemente por falta de memoria en el servidor de ***AWS***. Para solucionar el problema, se creó un ***Cron Job*** que mirase si el estado del proceso era activo ***(OK)***, cada diez minutos. En caso negativo, se volvía a ejecutar el script.

Por último, podríamos destacar la complejidad del perfil de usuario. Para no sobrecargar el servidor, algunas pestañas hacen peticiones al servidor mediante ***AJAX***, pero únicamente cuando el usuario interactúa con ellas.
También fue compleja la implementación de subida de "foto de perfil" dinámica, actualizando la foto sin refrescar la página, y poder subir la imagen arrastrando el archivo directamente en una zona resaltada de subida de archivo.

4.2.3.Insectos

En esta fase se implementó todo lo relacionado con la parte de insectos, desde la galería, hasta los mapas de distribución.

Para la galería, se tuvo que implementar un sistema de paginación que facilitará navegar libremente por todas las fotos subidas, ordenadas por orden de fecha de subida.

La subida de las fotos es compleja ya que está formada por un formulario que interactúa directamente con coordenadas GPS. Se integraron librerías de Google Maps para poder añadir coordenadas especificas del lugar de la captura, proporcionando a su vez un mapa que hace más fácil la interacción visual del usuario.

Los mapas de distribución se consiguen con una librería de JavaScript que añade nubes de colores (rojo más coincidencias, verde menos coincidencias) en el mapa, para así saber en qué zonas se ha capturado el insecto, y por lo tanto más probable encontrarlo en el futuro.
Las fotos subidas solo pasan a formar parte del mapa de distribución si un experto corrobora que es el insecto en cuestión, formando parte del perfil de ese insecto.
Para poder contar con datos reales y poblar así los mapas de distribución, se exportaron insectos usando una API externa proporcionada por ***GBIF***.
GBIF es una base de datos internacional que proporciona datos de biodiversidad de todo el mundo para apoyar la investigación científica, fomentar la conservación biológica y favorecer el desarrollo sostenible.
No todos los insectos han sido exportados, pero mediante un script se puede ejecutar una llamada a la API de ***GBIF*** y conseguir los últimos datos de insectos, añadiendo nuevas citas de captura para un insecto existente en nuestra base de datos, o añadiendo uno nuevo.

4.2.4.Chat

El chat es una pieza clave del proyecto, ya que proporciona una interacción social en tiempo real, haciendo la experiencia entre usuarios en la aplicación mucho más cercana.

Para implementar el chat se tuvo que crear una conexión bidireccional usando ***Websockets***, tanto en el lado del cliente como en el lado del servidor.
Uno de los retos fue mantener el chat operativo, incluso cuando el script de ***PHP*** que inicia la conexión en el lado del servidor se caía. Para conseguir esto se creó un Cron Job que reinicia el proceso cada cuatro horas, este activo o no, para asegurarnos que este operativo.
El lado del cliente se inicia mediante ***JavaScript***, y solo basta con acceder a cualquier página como usuario registrado. Se puede cerrar la conexión mediante un botón en la interfaz de usuario.

4.3.Migración a Docker e integración con Servicios Web de Amazon (EC2)

A medida que iba desarrollando la aplicación observé que tener que lidiar con la instalación, incompatibilidad de versiones, etc.. cuando necesitase instalar la aplicación en una nueva máquina iba a ser una pérdida de tiempo y recursos.
Llegué a la conclusión de que necesitaba una herramienta como Docker, que hiciese todo el trabajo por mí todas las veces que hiciese falta, pero yo solo tuviera que hacerlo una vez.

La adaptación de un proyecto ya empezado a Docker no es una tarea fácil y me llevo muchas horas de prueba y error, pero al final conseguí añadir los comandos necesarios en el ***Dockerfile*** para que funcionara correctamente. Aunque es verdad que necesite dos versiones, una para mi máquina local, y otra para la máquina local en AWS, que equivale a la versión de producción (*production* o *live*).

Básicamente lo que ***Docker*** hace es clonar el proyecto en el directorio escogido dentro de un entorno virtual y luego sigue una serie de pasos indicados previamente.
Estos pasos pueden ser los siguientes:

- Instalar la correcta versión de ***PHP*** y sus extensiones.
- Instalar y ejecutar ***Bower*** y ***Composer*** para instalar todas las dependencias que necesita el proyecto.
- Instalación de la correcta versión de ***Zend Framework.***
- Instalación del S***ervidor Apache.***
- Configuración de ***Cron Jobs.***
- Exponer puertos (para ***Websockets*** por ejemplo) **.**
- Dar permisos de usuarios y de propiedad a la carpeta del proyecto.

Una vez esto está funcionando, podemos añadir contenedores independientes para una mayor flexibilidad usando ***Docker Compose***.
Decidí tener tres contenedores: uno donde reside el código de la aplicación (y el principal), otro para RabbitMQ y otro para la base de datos MySQL, que está vinculado al principal.
El Dockerfile puede ser examinado en el anexo ***7. Dockerfile.***

El archivo ***docker-compose.yml*** sería así:

```
version: '3'
services:
  app:
    build:
      context: .
      dockerfile: docker/Dockerfile
    image: project-docker
    ports:
      - 80:80
      - 2020:2020
    volumes:
      - .:/srv/app
    links:
      - mysql
  mysql:
    image: mysql:5.7
    command: --sql_mode=""
    ports:
      - 13306:3306
    volumes:
    - ./database/dump.sql:/docker-entrypoint-initdb.d/dump.sql
    environment:
      MYSQL_ROOT_PASSWORD: dodgy
      MYSQL_DATABASE: foro
  rabbit1:
    image: "rabbitmq:3-management"
    hostname: "rabbit1"
    environment:
      RABBITMQ_ERLANG_COOKIE: "SWQOKODSQALRPCLNMEQG"
      RABBITMQ_DEFAULT_USER: "rabbitmq"
      RABBITMQ_DEFAULT_PASS: "rabbitmq"
      RABBITMQ_DEFAULT_VHOST: "/"
    ports:
      - "15672:15672"
      - "5672:5672"
    labels:
      NAME: "rabbitmq1"
```

Tabla 6. docker-compose.yml

Una vez terminado el proyecto, y teniendo docker funcionando correctamente en la máquina local, decidí integrar el proyecto a un servidor real de Amazon, así puede ser accedido mediante un navegador de internet.

La integración fue hecha en una instancia de ***EC2*** de los ***Servicios Web de Amazon (AWS)***, siguiendo los siguientes pasos:

- Crear una cuenta de ***AWS*** con acceso a la ***consola de administración.***
- Lanzar la instancia de ***EC2*** para crear y configurar la máquina virtual de Amazon:
  - Selección del sistema operativo, en este caso una imagen de Linux sin interfaz gráfica.
  - Selección del tipo y tamaño de la instancia, en este caso la más pequeña (***t2.micro***).
  - Configurar grupos de seguridad y usuarios para Linux.
  - Lanzar la instancia ***EC2***.
- Generar par de claves ***.pem.*** Esta clave se usará para acceder al servidor remotamente mediante ***SSH***. Ejemplo:

```
ssh -i ~/Desktop/TFG/insectworld.pem ubuntu@ec2-35-177-167-33.eu-west-2.compute.amazonaws.com
```

5.Integración, pruebas y resultados

5.1.Pruebas Unitarias (Unit Testing)

Las pruebas ***Prueba Unitarias o Unit Testing***, son unas piezas de código, pertenecientes a la metodología ágil, que se encargan de corroborar el correcto funcionamiento de la lógica de una pequeña parte del código.
Las principales características son las siguientes:

- Cada prueba debe ser ***rápida***, por lo que tiene que ser específica y abarcar la menor lógica posible.
- Las pruebas deben de ser ***independientes*** unas de otras, lo que quiere decir que el resultado de una prueba no debe afectar al resultado de otra. Esto se consigue simulando los datos de entrada en cada prueba.
- Las pruebas se deben de poder ***repetir***. Se van a tener que ejecutar cada vez que haya un cambio en nuestro código, lo que asegura que nada ha dejado de funcionar.
- Las pruebas tienen que ser claras a la hora de ***validar*** el resultado esperado.
- Se deben ejecutar las pruebas ***antes de producción*** para asegurar que nada está roto.

Las Pruebas Unitarias deben formar parte del proyecto ya que lo hace más fiable y robusto.
Las principales ventajas son las siguientes:

- El código en producción tiene muchos menos errores.
- Se pueden detectar bugs antes de que un usuario lo encuentre mientras usa la aplicación.
- Ayuda a tener una mayor comprensión del código, haciendo este más mantenible.
- Ayuda a descartar posibles problemas y errores, sabiendo que los tests están correctos, el error puede estar en otro sitio.
- Facilita las labores de mejora y refactorización del código, ya que te aseguras que tus cambios no han roto la lógica del código anterior.
- Se localizan los errores mucho más rápido, ya que las pruebas son localizadas.
- En general, se mejora la productividad, lo que conlleva una reducción de costes.

Para realizar las pruebas unitarias del proyecto utilice el framework ***PHPUnit,*** que es un framework de ***PHP*** que facilita la implementación de pruebas unitarias.
Funciona extendiendo la clase del test que se va a realizar a una de las clases que el framework proporciona. ***Zend Framework*** también proporciona sus propias clases para hacer pruebas unitarias, que a su vez extienden internamente a las clases de ***PHPUnit***.
Como el proyecto usa ***Zend Framework,*** que sigue una arquitectura ***MVC (Modelo-Vista-Controlador)***, no se han realizado pruebas en ninguna de las vistas, ya que no es necesario.

Tampoco se realizaron tests en los controladores, porque el propio framework ya ha realizado las pruebas pertinentes para asegurar su correcto funcionamiento.
Donde si es necesario realizar tests, y donde se han realizado, es en los ***Modelos***.
En los ***Modelos*** reside la lógica central del proyecto, ya que coge datos de la base de datos y aplica la lógica necesaria a estos.
Los tests se ejecutan desde la terminal, donde se ven los resultados del test.
Por ejemplo, para ejecutar el test de discusiones, se usaría el siguiente comando:

```
vendor/bin/phpunit UnitTest application/models/DiscusionModelTest.php
```

Los resultados pueden ser positivos o negativos, a continuación se dan dos ejemplos:

- Resultado positivo:

```
PHPUnit 8.5.3 by Sebastian Bergmann and contributors.

.....                                               5 / 5 (100%)

Time: 67 ms, Memory: 8.00 MB

OK (5 tests, 13 assertions)
```

Tabla 7. Test resultado positivo

- Resultado negativo:

```
PHPUnit 8.5.3 by Sebastian Bergmann and contributors.

F..F.                                               5 / 5 (100%)

Time: 64 ms, Memory: 8.00 MB

There were 2 failures:

1) DiscusionModelTest::testCheckNewMessagesInTopicAction
Failed asserting that 2 matches expected 1.

/Users/jose_toribio/projects/tfg/tests/application/models/DiscusionModelTest.php:18

2) DiscusionModelTest::testGetLastMessageAction
Failed asserting that false matches expected true.

/Users/jose_toribio/projects/tfg/tests/application/models/DiscusionModelTest.php:42

FAILURES!
Tests: 5, Assertions: 13, Failures: 2.
```

Tabla 8. Test resultado negativo

5.2.Pruebas de usuarios reales

Otras pruebas clave son las pruebas con usuarios reales utilizando la aplicación. Se le dejo usar la aplicación a diversas personas para que dieran un feedback sobre su experiencia. Entre ellas fueron:

- Mi padre, como usuario experto: al ser un gran conocedor de la materia, aportó nuevas ideas para mejorar las áreas más técnicas, como pueden ser los gráficos de los mapas de distribución.
- Usuarios menos familiarizados con la entomología, como inexpertos: proporcionaron información que ayudó a hacer la aplicación más amigable para los usuarios inexpertos.

En definitiva, las pruebas dieron como resultado una visión global del comportamiento de la aplicación en situaciones reales, que quizás el propio programador no habría notado, ya que conoce el código por debajo. Ayudó a destapar pequeños errores y bugs de baja importancia.

5.3.Pruebas visuales

5.3.1.Página de registro

Mirar anexo: ***1. Fotos de pruebas visuales de Registro***

5.3.2.Página de Inicio de sesión

Mirar anexo: ***2. Fotos de pruebas visuales de Inicio de Sesión***

5.3.3. Foro

Mirar anexo: ***3. Fotos de pruebas visuales del Foro***

5.3.4.Perfil

Mirar anexo: ***4. Fotos de pruebas visuales de Perfil De Usuario***

5.3.5.Sección determinar

Mirar anexo: ***5. Fotos de pruebas visuales de Determinar Insecto***

5.3.6.Chat

Mirar anexo: ***6. Fotos de pruebas visuales del Chat***

5.4. Mejoras y resultados

El uso de tests ha contribuido a una buen desarrollo y mantenimiento de partes ya codificadas del proyecto, evitando potenciales errores a la hora de añadir nuevas áreas de código.

Cuando implementaba nuevos fragmentos de código dentro de un código ya existente que estaba testado y funcionaba correctamente, me ha ayudado a localizar fallos en la lógica de ese código, pudiendo hacer las correcciones necesarias directamente en caso de que el test fallara.

También me ha ayudado a afianzar la lógica usada en diferentes áreas, ya que el test debe de seguir estrictamente una lógica determinada, haciéndome profundizar en el funcionamiento de la aplicación.

Otra pieza clave de los tests es que me ayudó a identificar vulnerabilidades en el código, como uso de variables de entrada no comunes que puedan romper la aplicación o causar alguna entrada o borrado de información sensible en la base de datos.

Por último, se puede barajar la opción de mejorar algunas áreas de la aplicación:

- Dar la opción de utilizar la localización actual automáticamente cuando se suba una foto de captura nueva.
- Poder utilizar el chat en una ventana externa.
- Integrar plugins para visualizar imágenes.
- Listado global de insectos junto con sus mapas de distribución.
- Mejorar el sistema de paginación en la galería.
    - Más flexibilidad de navegación entre páginas.
    - Utilizar un sistema de cache para cargar más rápido y no sobrecargar el servidor.

Podría añadir más mejoras, ya que un proyecto nunca deja de poder mejorar con nuevas tecnologías y con el mantenimiento, pero esas son algunas de las que más ayudarían en mi opinión.

6.Conclusiones y trabajo futuro

6.1.Conclusiones

Desarrollar este proyecto ha sido una experiencia que me ha ayudado a crecer profesional y personalmente. Ha puesto a prueba mis conocimientos, capacidad de resolución de problemas y búsqueda de soluciones. Las situaciones a las que me he enfrentado son situaciones que se asemejan a las que se pueden encontrar en el ámbito laboral, por lo que ha sido una buena forma de coger experiencia y soltura para el futuro.

Me ha servido también para darme cuenta de mis errores y mis puntos débiles, pudiendo así profundizar en ellos en un futuro cercano. Me ha quedado clara la importancia de planificar a priori y no adentrarse a escribir el código sin tener primero una idea clara de lo que se quiere hacer y los posibles problemas a los que uno se puede enfrentar.
Leer y entender la documentación de terceras partes usadas en el proyecto es también muy importante para no malgastar tiempo descubriendo cómo funcionan.

En definitiva, me ha servido para darme cuenta de la importancia que tiene nuestra profesión en cualquier ámbito de la vida, ya que tiene una influencia transversal, abarcando muchísimos campos diferentes. Aunque la gente no siempre sea consciente de ello, la informática es una parte esencial de sus vidas, y como en este proyecto, ayuda a juntar a gente con inquietudes e intereses similares.

6.2.Trabajo futuro

Una vez terminado este proyecto, puedo centrarme en mejorarlo, o iniciar un nuevo proyecto aplicando el conocimiento adquirido.

Las mejoras incluirían una actualización de la tecnología que lo necesitara, nuevas herramientas y adaptar la aplicación a dispositivos móviles. Esta última es la más interesante, ya que abarcaría a más usuarios y aprendería sobre el desarrollo de aplicaciones para teléfonos móviles.

Un nuevo proyecto también es interesante porque puedo usar y aprender cualquier tecnología que me apetezca. Aun no tengo en mente ninguna idea para una posible nueva aplicación.

Referencias

1. RabbitMQ Server Documentation, v3.8.3. Accessed on: February 21, 2020. [Online]. Available: https://www.rabbitmq.com/documentation.html
2. Zend Framework Documentation, v1.12. Accessed on: October 14, 2019. [Online]. Available: https://framework.zend.com/manual/1.12/en/manual.html
3. PHP Documentation, v7.3. Accessed on: October 18, 2019. [Online]. Available: https://www.php.net/docs.php
4. W3Schools Tutorials, Accessed on: January 28, 2020. [Online]. Available: https://www.w3schools.com
5. MYSQL Documentation, v8.0, Accessed on: November 30, 2019. [Online]. Available: https://dev.mysql.com/doc
6. LAMP Stack Installation, Accessed on: November 26, 2019. [Online]. Available: https://help.ubuntu.com/lts/serverguide/lamp-overview.html
7. Docker Start Guide, v19.03.5, Accessed on: March 07, 2020. [Online]. Available: https://docs.docker.com/get-started/
8. Websockets Client Documentation, v1.3.0, Accessed on: March 28, 2020. [Online]. Available: https://socket.io/docs/
9. Websockets Server Documentation, v1.3.0, Accessed on: March 28, 2020. [Online]. Available: https://awesomeopensource.com/project/walkor/phpsocket.io
10. API GBIF, v1, Accessed on: January 15, 2020. [Online]. Available: https://www.gbif.org/developer/summary
11. AWS EC2 Documentation, Accessed on: April 10, 2020. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
12. PHPUnit Documentation, v8.1, Accessed on: October 30, 2019. [Online]. Available: https://phpunit.readthedocs.io/en/8.1/

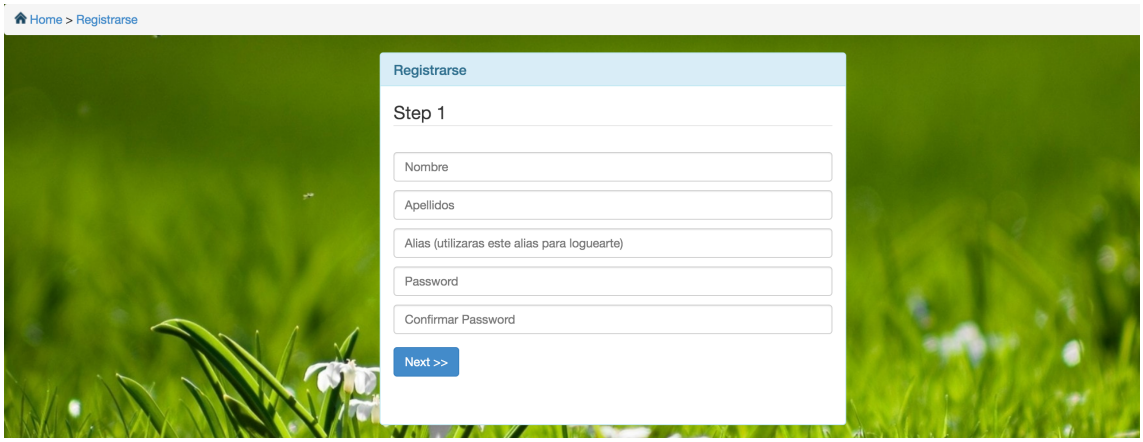
Glosario

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASP	Active Server Pages
AWS	Amazon Web Services
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheet
EC2	Elastic Computer Cloud
GBIF	Global Biodiversity Information Facility
GPS	Global Positioning System
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
JSP	Java Server Page
LAMP	Linux, Apache, MySQL, and PHP
MVC	Model-View-Controller
ORM	Object-Relational Mapping
PHP	Hypertext Preprocessor
POO	Programación Orientada a Objetos
PSR	PHP Standard Recommendation
RSA	Rivest–Shamir–Adleman
SSH	Secure Shell

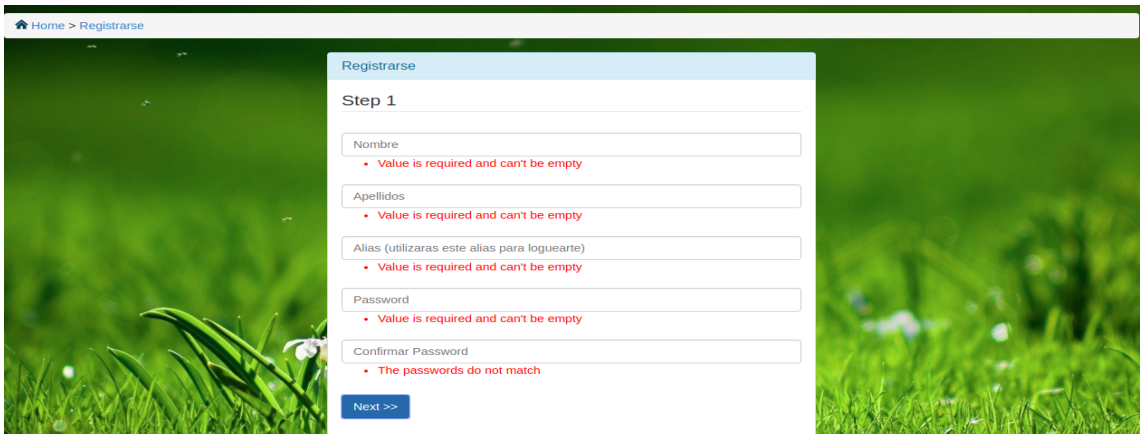
Anexos

1. Fotos de pruebas visuales de Registro

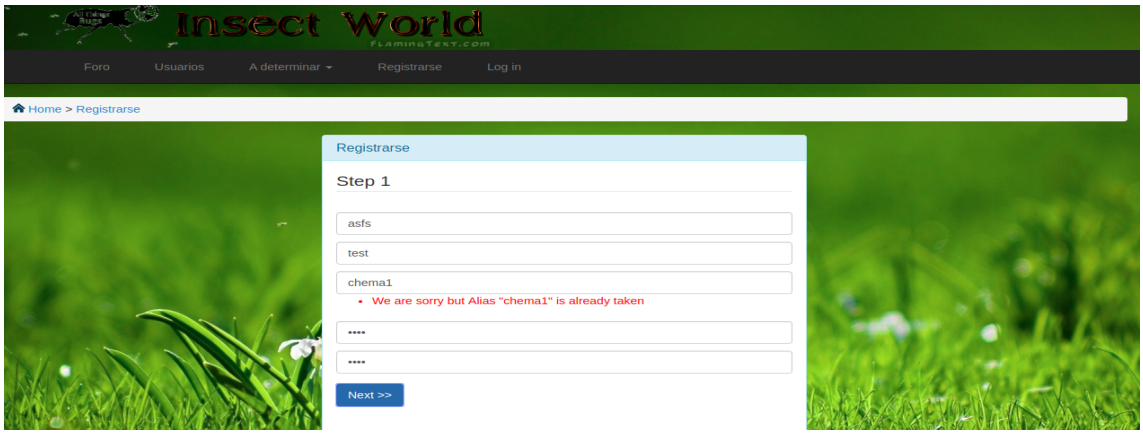
- Primer formulario de la página de registro:



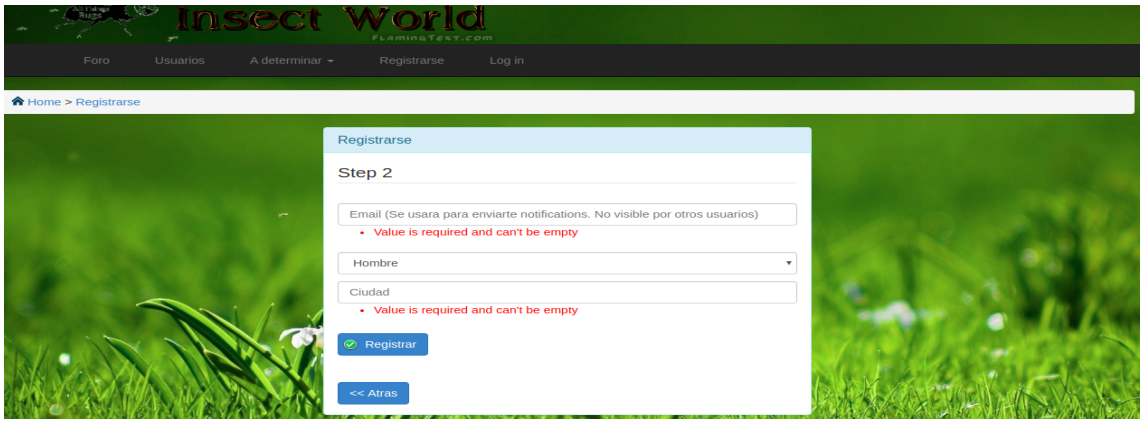
- Primer formulario de la página de registro con errores (1) :



- Primer formulario de la página de registro con errores (2) :

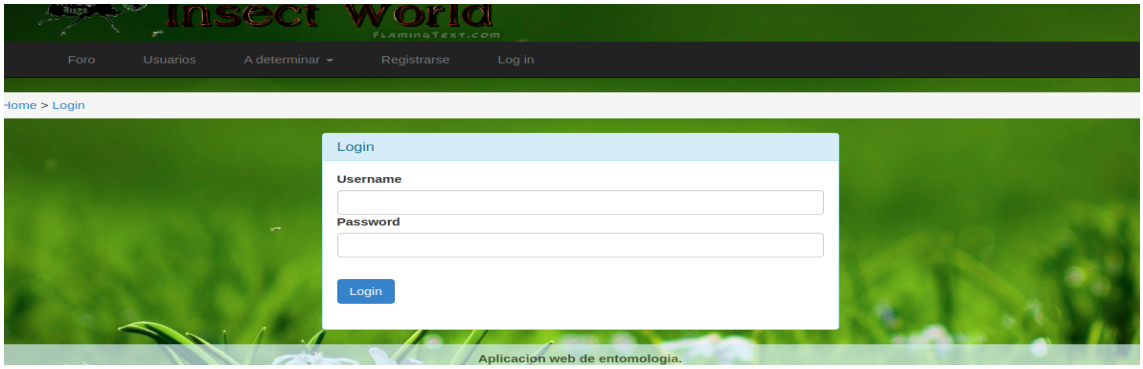


- Segundo formulario de la página de registro con errores:

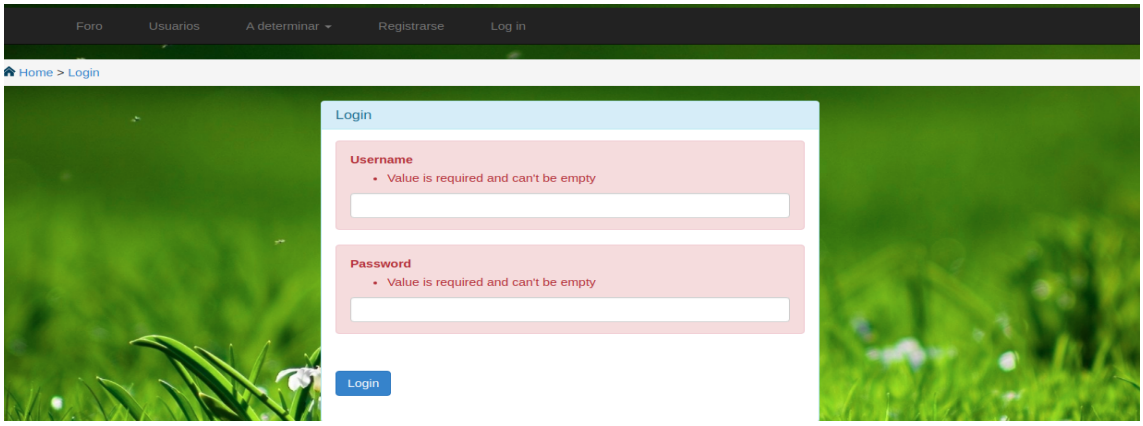


2. Fotos de pruebas visuales de Inicio de Sesión

- Segundo formulario de la página de registro con errores:



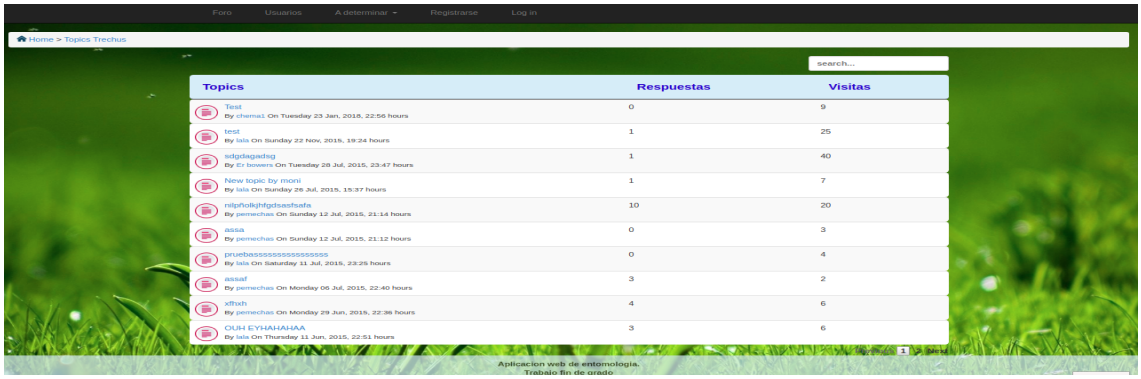
- Segundo formulario de la página de registro con errores:



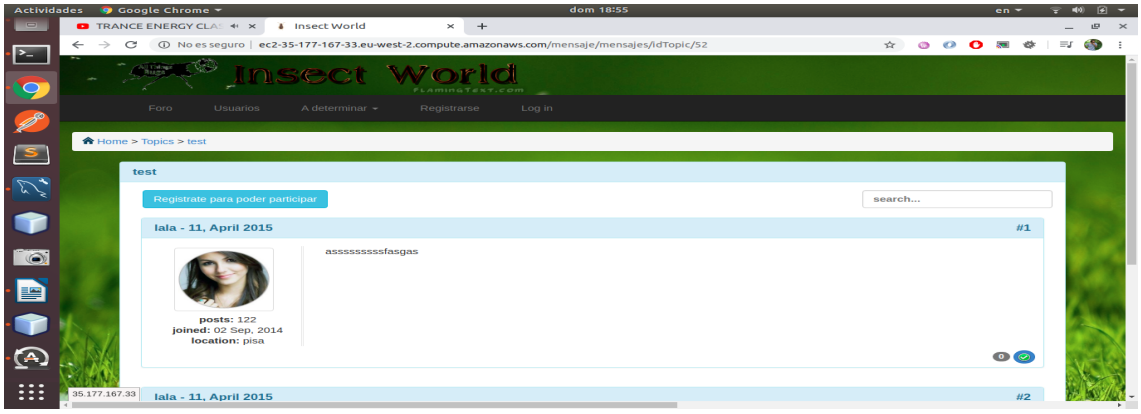
3. Fotos de pruebas visuales del Foro

- Discusiones del foro:

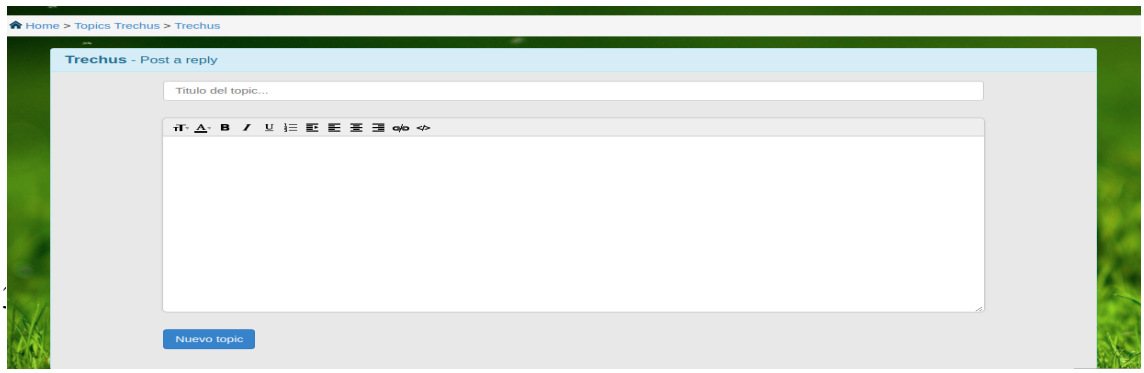
- Topics del foro:



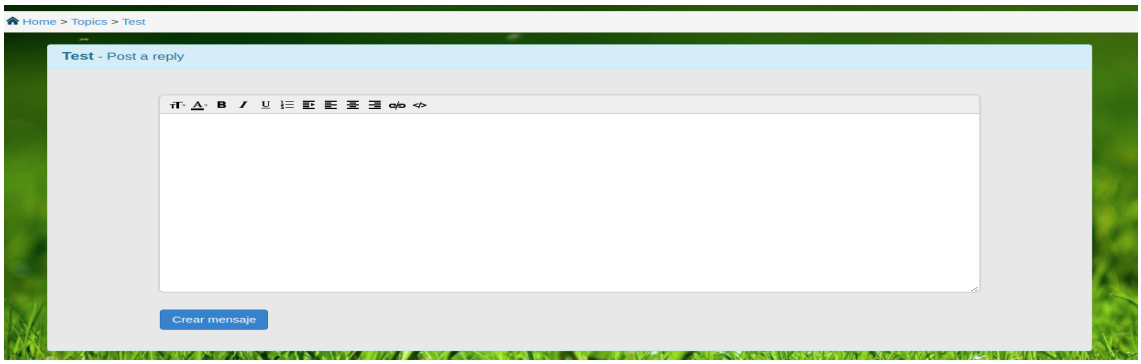
- Mensajes del foro:



- Crear nuevo topic:



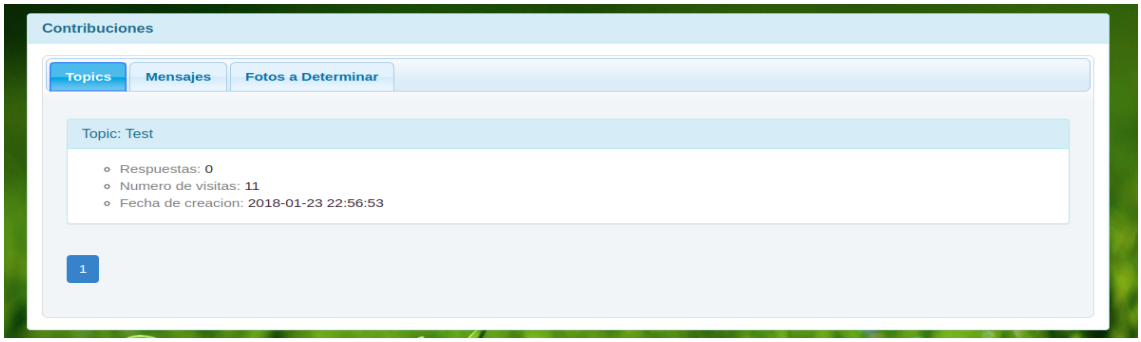
- Crear nuevo mensaje:



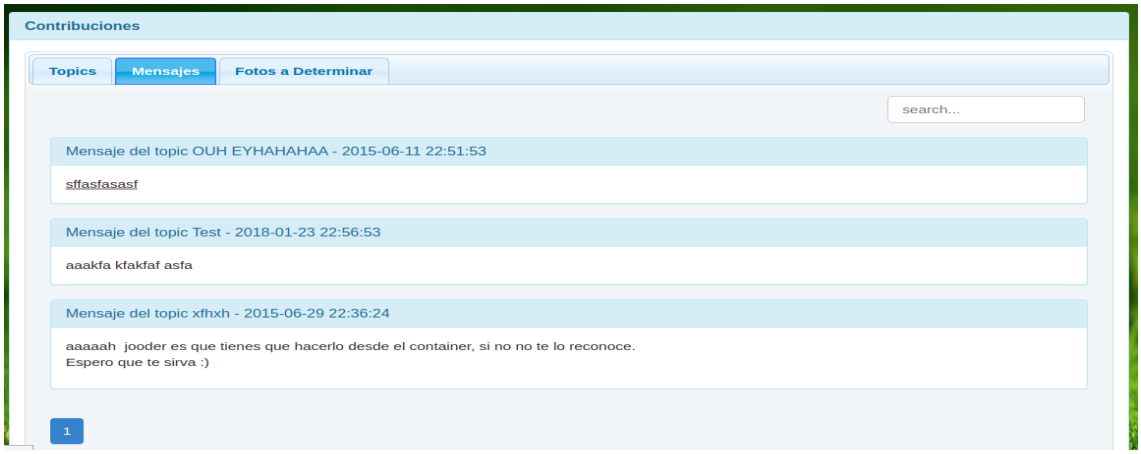
4.Fotos de pruebas visuales de Perfil De Usuario

- Información personal:

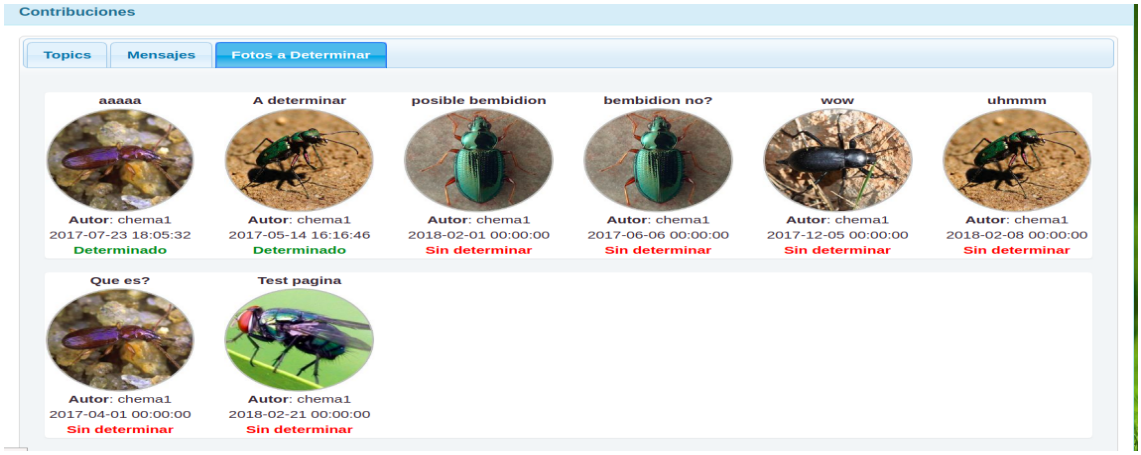
- Contribución topics del foro:



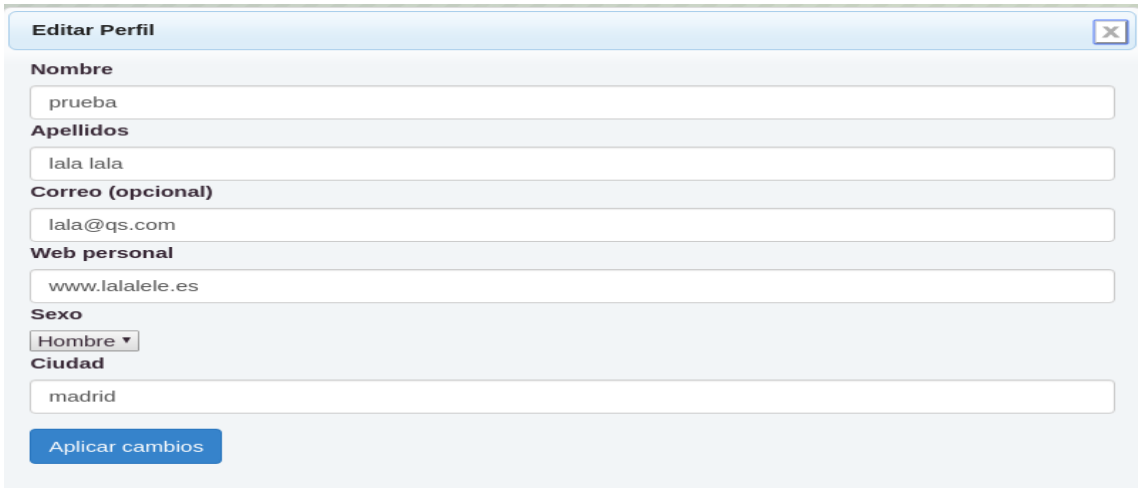
- Contribución mensajes del foro:



- Contribución fotos a determinar:



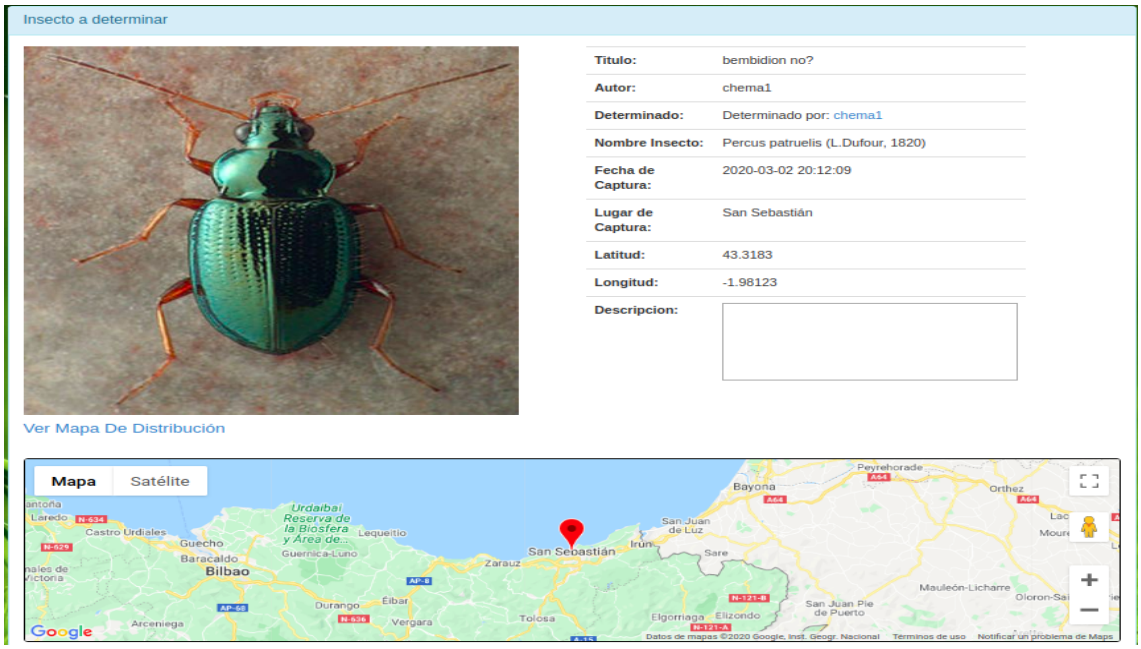
- Editar información personal:



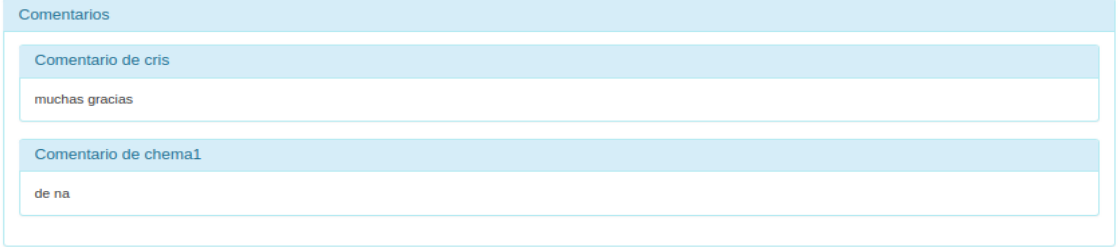
5.Fotos de pruebas visuales de Determinar insecto

- Galería de insectos:

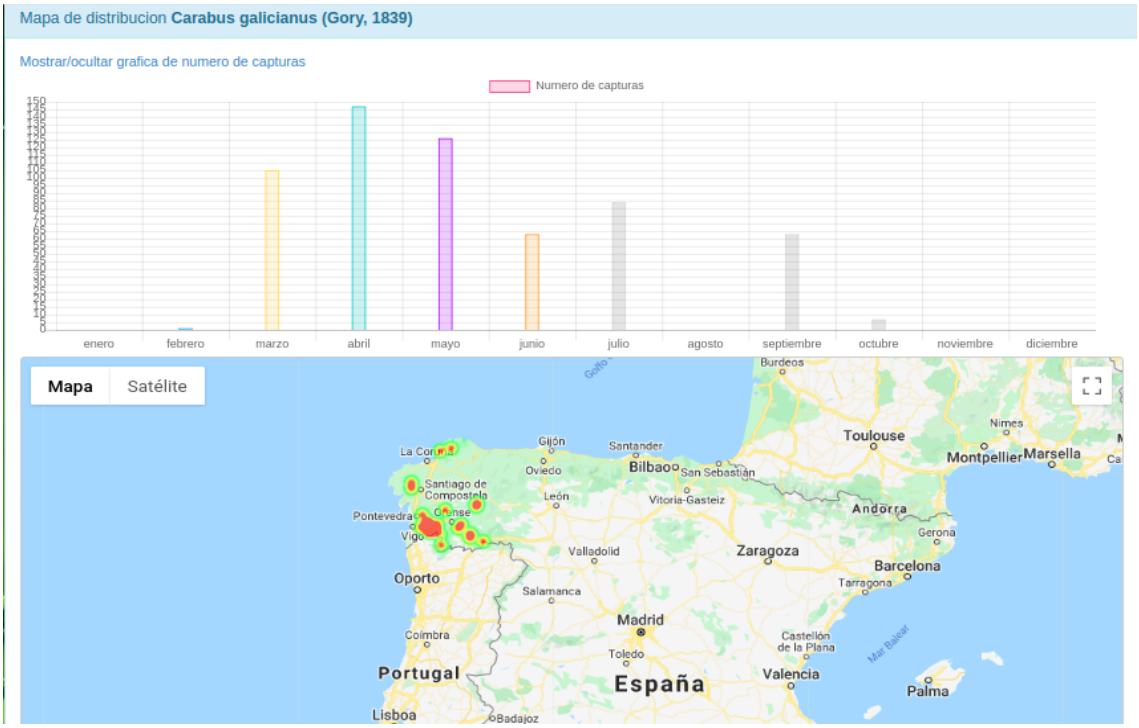
- Perfil de insecto:



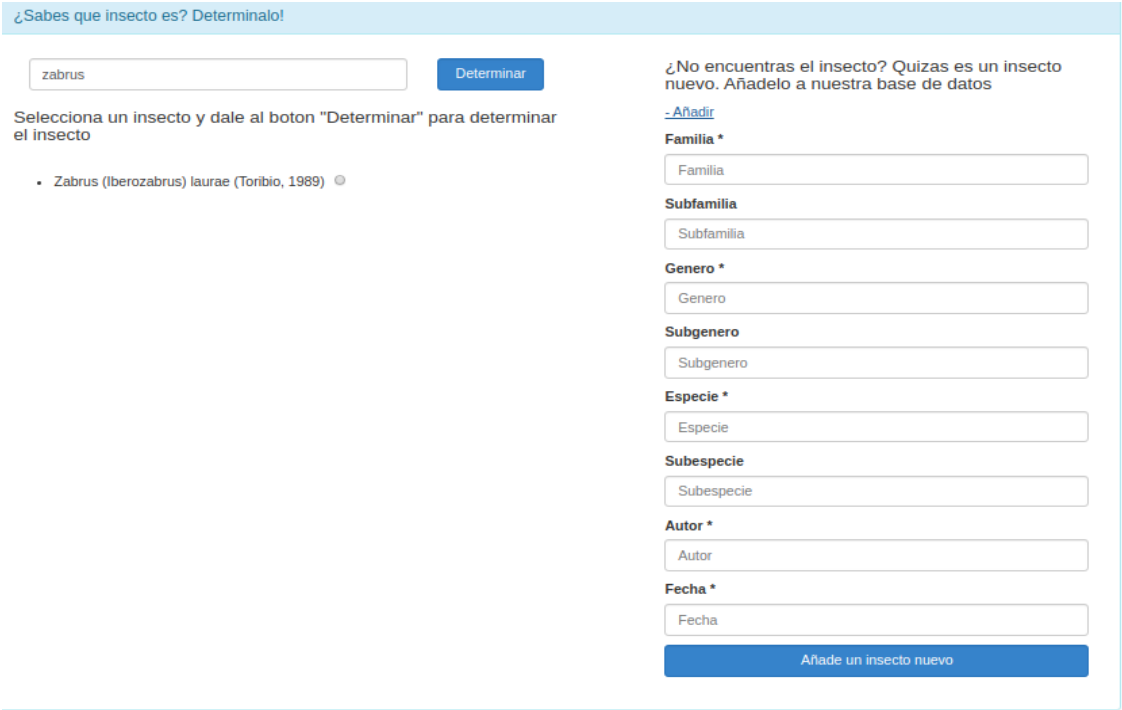
- Perfil de insecto comentarios:



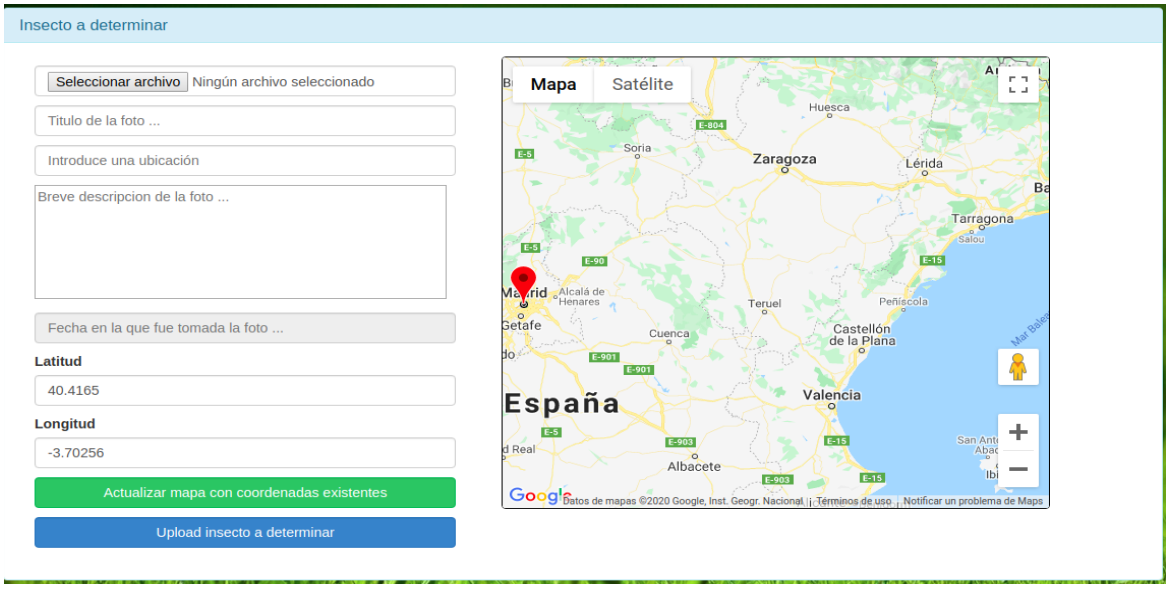
- Mapa de distribución y estadísticas:



- Insecto no determinado con búsqueda y permisos para determinar:

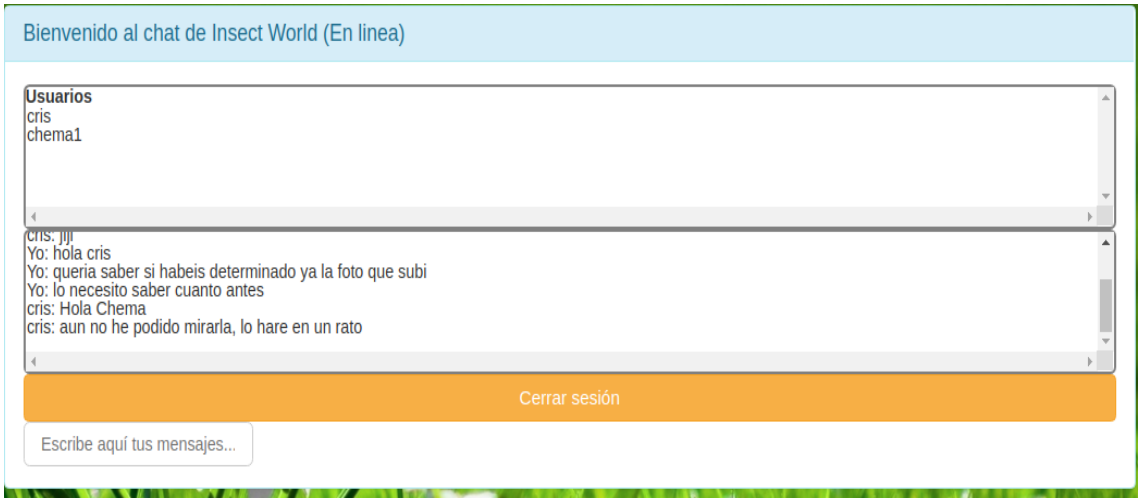


- Formulario insecto a determinar:

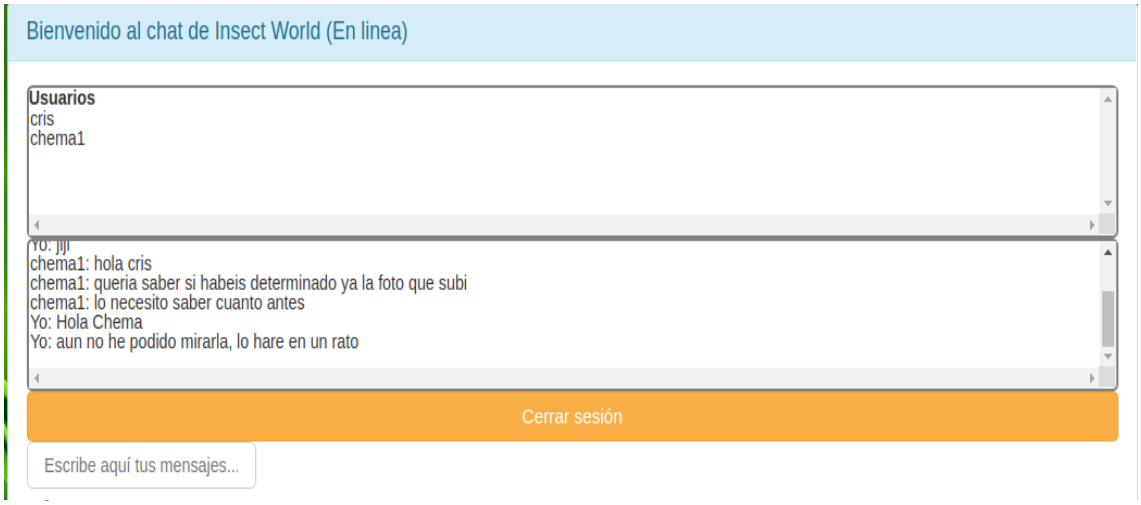


6.Fotos de pruebas visuales del Chat

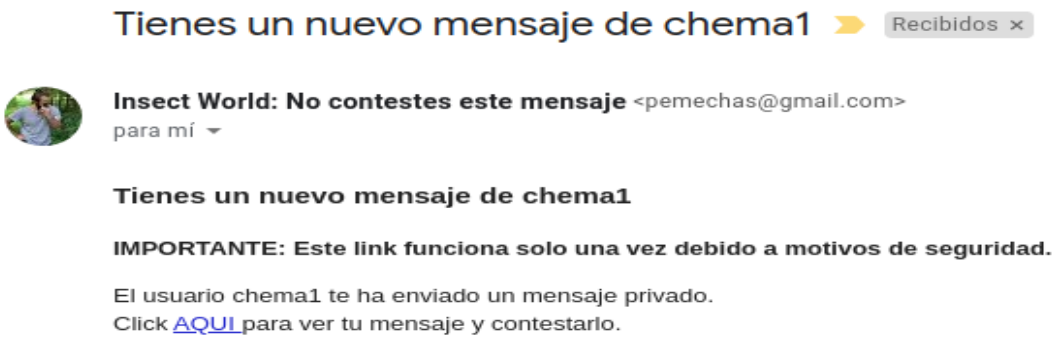
- Usuario Chema1 en el chat:



- Usuario Cris en el chat:



- Email automático con rabbitMQ nuevo mensaje:



El link the autologueo solo funciona una vez.

7.Dockerfile

```
FROM php:7.1.8-apache

MAINTAINER Paul Redmond

COPY . /srv/app
COPY docker/vhost.conf /etc/apache2/sites-available/000-default.conf

RUN a2enmod rewrite
RUN a2enmod headers expires
RUN service apache2 restart

RUN apt-get update
RUN apt-get install -y --no-install-recommends
RUN apt-get install -y -qq git curl wget
RUN apt-get install -y vim cron

# Set up cronjobs
RUN cp /srv/app/cronjobs/crontab /etc/cron.d/cronchat

# install npm
RUN apt-get install -y -qq npm
RUN ln -s /usr/bin/nodejs /usr/bin/node

# install bower
RUN npm install --global bower

RUN wget -P /srv/ "https://packages.zendframework.com/releases/ZendFramework-1.12.9/ZendFramework-1.12.9.tar.gz"
RUN cd /srv/ && tar -zxf ZendFramework-1.12.9.tar.gz
RUN rm /srv/ZendFramework-1.12.9.tar.gz
RUN rm -R /srv/app/library/Zend
RUN ln -s /srv/ZendFramework-1.12.9/library/Zend /srv/app/library/Zend

WORKDIR /srv/app

# Install php extensions, composer
RUN docker-php-ext-install mbstring mysqli pcntl
RUN docker-php-ext-enable mysqli
RUN docker-php-ext-install bcmath
RUN curl --silent --show-error https://getcomposer.org/installer | php
RUN php composer.phar update

RUN bower install --allow-root

EXPOSE 2020

# Update existing symlinks from bower
RUN rm /srv/app/public/js/plugins/moment
RUN ln -s /srv/app/bower_components/moment /srv/app/public/js/plugins/moment
RUN rm /srv/app/public/js/plugins/chart.js
RUN ln -s /srv/app/bower_components/chart.js /srv/app/public/js/plugins/chart.js
RUN rm /srv/app/public/js/plugins/list.pagination.js
RUN ln -s /srv/app/bower_components/list.pagination.js /srv/app/public/js/plugins/list.pagination.js
RUN rm /srv/app/public/js/plugins/list.js
RUN ln -s /srv/app/bower_components/list.js /srv/app/public/js/plugins/list.js
#RUN rm -R public/js/plugins/jquery-file-upload/
RUN ln -s /srv/app/bower_components/jquery-file-upload /srv/app/public/js/plugins/jquery-file-upload
#RUN ln -s /srv/app/bower_components/jquery-knob/dist/jquery.knob.min.js /srv/app/public/js/plugins/jquery-file-upload/js/vendor/
jquery.knob.min.js

RUN chown -R www-data:www-data /srv/app

ENV APP_ENV=production

# 1- cp /usr/local/etc/php/conf.d/docker-php-ext-mysqli.ini /usr/local/etc/php/conf.d/php.ini
# 2- add to php.ini mysqli.default_socket = "/var/run/mysqld/mysqld.sock"
```

Tabla 9. Dockerfile